



# **SANDIA REPORT**

SAND2002-0245

Unlimited Release

Printed February 2002

## **Pattern Discovery in Time-Ordered Data**

Gregory N. Conrad, John M. Britanik, Sharon M. DeLand, and Christina L. Jenkin

Prepared by  
Sandia National Laboratories  
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,  
a Lockheed Martin Company, for the United States Department of  
Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



**Sandia National Laboratories**

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from  
U.S. Department of Energy  
Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831

Telephone: (865)576-8401  
Facsimile: (865)576-5728  
E-Mail: [reports@adonis.osti.gov](mailto:reports@adonis.osti.gov)  
Online ordering: <http://www.doe.gov/bridge>

Available to the public from  
U.S. Department of Commerce  
National Technical Information Service  
5285 Port Royal Rd  
Springfield, VA 22161

Telephone: (800)553-6847  
Facsimile: (703)605-6900  
E-Mail: [orders@ntis.fedworld.gov](mailto:orders@ntis.fedworld.gov)  
Online order: <http://www.ntis.gov/ordering.htm>



# **Pattern Discovery in Time-Ordered Data**

Greg N. Conrad, John M. Britanik, Sharon M. DeLand, and Christina L. Jenkin  
Advanced Decision Support Applications Department  
Sandia National Laboratories  
P.O. Box 5800  
Albuquerque, New Mexico 87185-1137

## **Abstract**

This report describes the results of a Laboratory-Directed Research and Development project on techniques for pattern discovery in discrete event time series data. In this project, we explored two different aspects of the pattern matching/discovery problem. The first aspect studied was the use of Dynamic Time Warping for pattern matching in continuous data. In essence, DTW is a technique for aligning time series along the time axis to optimize the similarity measure. The second aspect studied was techniques for discovering patterns in discrete event data. We developed a pattern discovery tool based on adaptations of the A-priori and GSP (Generalized Sequential Pattern mining) algorithms. We then used the tool on three different application areas – unattended monitoring system data from a storage magazine, computer network intrusion detection, and analysis of robot training data.

# Acknowledgments

The authors wish to thank the Emerging Threat and Non-Proliferation and Material Control Strategic Business Units for their joint sponsorship of this LDRD project. We also wish to thank Bobby Corbell for sensor data from a simulated storage facility and numerous discussions about its interpretation. We are thankful to Kevin Nauer and Chris Forsythe for data (network intrusion detection systems and robot training sequences, respectively) and feedback on many of the ideas presented in this paper. We are grateful to Eamonn Keogh for sample implementation code and extensive advice on Dynamic Time Warping. We are also grateful to Tim Oates for his extensive discussion of Hidden Markov Models and a comparison of the Dynamic Time Warping and Hidden Markov Model algorithms. Finally, we wish to thank John Brabson and John Mitchiner for their continuing interest and support of this project.

# Contents

Acknowledgments.....	4
Acronyms and Abbreviations .....	8
1 Introduction.....	9
2 Related Work .....	11
2.1 Pattern Discovery in Scientific Data .....	11
2.1.1 Hierarchical Clustering .....	11
2.1.2 Pattern Discovery.....	12
2.2 AVATAR.....	12
2.2.1 User-Driven Pattern Discovery.....	12
2.2.2 Learning .....	13
2.2.3 Pattern Detection.....	13
2.3 Feature Characterization in Scientific Datasets .....	13
2.3.1 Data Format.....	13
2.3.2 Features as Patterns .....	14
2.3.3 Framework Algorithms .....	14
2.4 VxInsight.....	14
2.4.1 Computing Object Similarity.....	15
2.4.2 Navigating Science .....	15
2.4.3 Moving Forward .....	16
3 Dynamic Time Warping and Hidden Markov Models .....	17
3.1 Dynamic Time Warping.....	17
3.1.1 An Example of DTW .....	19
3.1.2 Problems with DTW .....	20
3.2 Hidden Markov Models .....	22
3.2.1 Advantages and Disadvantages of HMMs.....	22
3.3 Directions for Further Work .....	23
4 Pattern Discovery in Discrete Event Time Series Data .....	24
4.1 Introduction.....	24
4.2 Event Tokenization.....	24
4.3 Preprocessing the Sequence .....	25
4.4 Pattern Discovery.....	26
4.4.1 Related Efforts .....	26
4.4.2 Adaptations to A-priori and GSP .....	27
4.4.3 Pattern Significance .....	27
4.4.4 Java Implementation.....	28
4.4.5 Future Investigation in Sequence Pattern Discovery.....	28
4.5 Post Processing and Analysis.....	28
4.5.1 Elementary Patterns .....	28
4.5.2 Noise Avoidance .....	29
4.5.3 Statistical Measures.....	30
4.6 String Likeness Measures .....	31
4.6.1 Human Interpretation of Likeness.....	31
4.6.2 A World of Choices .....	32
4.6.3 Related Work in String Similarity Assessment .....	32
4.6.4 Adaptations for Our Needs .....	33

4.6.5	Conclusion .....	36
5	The Pattern Discovery Tool.....	37
5.1	Introduction.....	37
5.2	Tool Components .....	37
5.3	Editing Sequences.....	38
5.4	Pattern Statistics .....	39
5.5	Table Sorting.....	39
5.6	Pattern Information.....	40
5.7	Pattern List Functions .....	42
5.8	Applications .....	43
5.8.1	Unattended Monitoring Data .....	43
5.8.2	Network Intrusion Detection.....	50
5.8.3	Robot Time Sequences.....	51
6	Summary and Future Work.....	53
7	References .....	54
8	Appendix: String Similarity Poll .....	57

## List of Figures

Figure 1:	Capturing user-discovered patterns .....	12
Figure 2:	Comparison of similarity measure based on Euclidean distance and DTW. ...	18
Figure 3:	An example of a warping path. ....	19
Figure 4:	Example time series.....	19
Figure 5:	The $n$ -by- $m$ matrix with the warping path highlighted.....	20
Figure 6:	Example where DTW does not find the obvious, natural alignment .....	21
Figure 7:	Comparison of DTW and DDTW .....	21
Figure 8:	The main <i>Sequence Pattern Discovery Tool</i> screen .....	38
Figure 9:	The <i>Edit</i> drop down menu .....	39
Figure 10:	The <i>Statistics</i> drop down menu .....	39
Figure 11:	The <i>Sort Table</i> drop down menu.....	40
Figure 12:	The <i>Info</i> drop down menu .....	40
Figure 13:	The <i>Patterns Like ...</i> window .....	41
Figure 14:	The <i>Regular Expression Entry</i> dialogue .....	41
Figure 15:	The <i>Patterns Like ...</i> window -- user provided regular expression .....	42
Figure 16:	The <i>Pattern Filter Options</i> window.....	43
Figure 17:	The main <i>Sequence Pattern Discovery Tool</i> screen for bunker analysis .....	45
Figure 18:	The updated main <i>Sequence Pattern Discovery Tool</i> screen .....	47

## List of Tables

Table 1:	Bunker sensor event type tokens .....	24
Table 2:	Patterns for sequence ABXABZ.....	27
Table 3:	A sample of patterns containing the possible elementary pattern CE .....	29
Table 4:	Token noise and absence analysis .....	33
Table 5:	Penalty component weighting by token.....	34
Table 6:	Offset calculations between ABCA and BBAD.....	35
Table 7:	Penalty component weighting.....	35

Table 8: Event tokenization scheme for initial analysis .....	43
Table 9: Event tokenization scheme for login process analysis .....	48
Table 10: Frequency of selected normal login patterns .....	49
Table 11: Frequency of selected login patterns .....	49
Table 12: Runs within the data set.....	51
Table 13: Frequent patterns found across robot runs .....	52

# Acronyms and Abbreviations

ASCI	Accelerated Strategic Computing Initiative
DDTW	Derivative Dynamic Time Warping
DOE	Department of Energy
DTW	Dynamic Time Warping
FCDMF	Feature Characterization using Data Models and Formats
GSP	Generalized Sequential Pattern
HMM	Hidden Markov Model
LDRD	Laboratory-Directed Research and Development
MF	Measured Frequency
MF	Measured Probability
NWC	Nuclear Weapons Complex
RF	Radio Frequency
SAF	Set and Fields
SGI	Silicon Graphics, Inc.
SOH	State-of-Health



# 1 Introduction

This report describes the results of a Laboratory-Directed Research and Development project which explored techniques for pattern discovery in discrete event time series data. The original motivation for the project was based on observed challenges in analyzing data from network-based sensor-driven monitoring systems. An example of such a system is one of the unattended monitoring systems fielded by the International Security Programs Center to monitor the status of high value assets and processes, particularly with respect to international nuclear material safeguards, nonproliferation, and transparency. In these applications, the purpose of the monitoring system may include detecting intrusion into a secured area, verification that known processes are occurring as expected, and detection of diversion of nuclear material. Another example of such a system is a network intrusion detection system used to monitor computer network communication traffic and user sessions.

Analysis of data from such systems requires identifying and classifying patterns in the sensor data and interpreting them in terms of the expected activities or allowed activities. A key issue from an operational perspective is that it is not feasible to have a human perform all of the analysis. A simple approach to automating the analysis is to identify patterns, classify the patterns as normal (or allowed, expected, etc.) and abnormal (or suspicious, unauthorized, etc.), and use pattern-matching algorithms to identify and classify observed behavior. Most network intrusion detection tools follow this approach. A finite state-machine based pattern detection approach has been successfully implemented as the Knowledge Generation software tool and demonstrated in nuclear material safeguards and transparency applications. [1][2][3] However, pattern detection tools generally cannot discover new, unknown patterns in the data. New tools are needed that are capable of pattern discovery – that is the identification of new patterns in the data whether those patterns represent normal activity or whether they are indicative of unauthorized, anomalous activities.

*Data mining* is a process for finding useful information from large data sets that involves a collection of algorithms and techniques for finding and categorizing patterns in data. [4][5] The techniques fall into several broad categories including *rule discovery*, the testing of patterns hypothesized by a user; *rule induction*, the automated extraction of patterns in the data; *regression*, the detection of patterns in continuous data; *deviation detection*, the detection of deviations from established or normal behavior; and *classification*, the categorization of data records or patterns in the data. [4] In essence, the pattern discovery problem we addressed in this LDRD is a data mining problem. While each of the categories listed above is important for automating analysis from unattended monitoring systems, we focused our efforts primarily on rule induction and rule discovery techniques.

In common commercial applications, data mining is directed toward discovering associations between attributes of an object and selecting associations which are useful in predicting some desirable or undesirable outcome. The “dimensionality” of the data refers to how many attributes (fields) are available for each object (record). Sensor data differ from commercial enterprise data in three significant ways:

1. *The data are of low-dimensionality.* Sensor data generally consist of the time of the event and a value for the event. While spectral and hyperspectral data can have high dimensionality, simpler sensors (i.e., temperature, voltage, etc.) do not.
2. *Some data may be continuous rather than discrete.* Data mining techniques generally apply to discrete data, so methods for discretizing the continuous data are needed. One approach is to bin the data. Another approach is to segment the data and flag the segment end-points as potential key events. We favor the latter approach since it matches the discrete event nature of the other sensors in the monitoring systems.
3. *The time ordering of the data is significant.* The pattern discovery techniques will need to extract information from data based not only on its order, but also based on the delay between events – considering both time-out as well as time-in events.

As a result of these differences, commercial data mining applications are not suited for finding sequence-based patterns in time-ordered data, although commercial applications can be useful for characterizing normal behavior based on time of day or day of week.

After initial exploration of the capabilities of commercial systems, this LDRD focused on developing algorithms for discovering sequence-based patterns. The test data we had contained data from both discrete sensors (e.g., door switches or breakbeams) and analog sensors reporting continuous-valued quantities (e.g., radiation levels or temperature). The different types of data (continuous vs. discrete) require different algorithms. As a result, there were two major thrusts to the LDRD. The first aspect studied was the use of Dynamic Time Warping (DTW) [6][7][8][9][10] for pattern matching in continuous data. In essence, DTW is a technique for aligning time series along the time axis to optimize the similarity measure. The second problem was the discovery of patterns in discrete event data. We developed a Pattern Discovery Tool based on adaptations of the A-priori [11] and GSP (Generalized Sequential Pattern mining) [12] algorithms. We then used the tool on three different application areas – unattended monitoring system data from a storage magazine, network intrusion detection, and analysis of robot training data.

The report is organized as follows: Chapter 2 describes other data mining work, particularly in the DOE complex. Chapter 3 describes the DTW algorithm and its application to pattern matching with continuous data. It also compares the DTW algorithm to another technique for describing time series data, Hidden Markov Models. Chapter 4 describes the general approach to pattern discovery in sequential data while Chapter 5 describes the Pattern Discovery tool and its applications.

## 2 Related Work

In this section, we examine several relevant major efforts in data mining at three DOE laboratories in the nuclear weapons complex (NWC).

High-resolution simulations, such as those used in the ASCI program, can generate terabytes of scientific data, which is much too voluminous for a human to analyze by hand. Several efforts within ASCI seek to explore data mining techniques to sift through the data and isolate the smaller, more manageable portions of interest for detailed analysis. Tools from these efforts analyze the data by identifying patterns, either automatically, or through interaction with the user. A pattern in the ASCI context can be anything from a sequence of “interesting” regions in mesh data to statistical similarities between characteristics of regions of interest to recurring relations among arbitrarily complex objects in the domain. We describe three of these data mining efforts below, namely Pattern Discovery in Scientific Data, AVATAR, and Feature Characterization in Scientific Datasets. In addition, below we describe related efforts in computer network intrusion detection, where it is desirable to discover anomalous patterns in network traffic, and we describe the VxInsight effort which graphically displays data in a geometric vicinity based on similarity metrics.

### ***2.1 Pattern Discovery in Scientific Data***

Karypis and Kumar [13] are developing clustering algorithms and pattern discovery approaches to operate on turbulent fluid flow and structural mechanics simulations. In their approach, higher order objects that correspond to interesting structures (such as vortices in flow simulation data) obtained from feature extraction are analyzed by the clustering and pattern discovery algorithms. The results of this analysis provide the user with high-level information that will assist in the processing and understanding of the key relationships in the simulation data. Of particular relevance to this LDRD project is the work in the discovery of frequent patterns.

#### **2.1.1 Hierarchical Clustering**

Clustering scientific data can be challenging due to the large volume of high-dimensional data. In addition, existing clustering algorithms are designed to use fixed metrics that limit their applicability to scientific datasets. For example, the K-means algorithm uses a fixed distance metric to cluster data around centroids. Such a metric is inadequate if one needs to cluster higher-order features such as vortices or cracks. To address this problem, recent work in hierarchical agglomerative clustering can be applied using dynamic metrics to measure the similarity between clusters. The use of dynamic metrics facilitates the discovery of natural and homogenous clusters of high-level objects in scientific data. A drawback of hierarchical agglomerative clustering is that the runtime tends to be  $O(n^2)$ , which can be unacceptable given large scientific datasets. This work investigates an approach based on data summarization [14] to limit the volume of data to be clustered while still yielding high-quality clusters, leading to a scalable solution.

### 2.1.2 Pattern Discovery

The discovery of frequent (or infrequent) patterns among objects in the dataset is an important problem in data mining. Karypis and Kumar[13] define a pattern as a recurring relation among objects, with an importance determined by how frequently it occurs (support level) and how indicative its occurrence is of a certain outcome (confidence level). Patterns that have a high confidence level are very important because they can provide an accurate prediction. The support of a pattern is also important, as patterns that do not occur frequently may be spurious. On the other hand, Karypis and Kumar[13] note that sometimes infrequent patterns can be very important, and the challenge is to distinguish infrequent important patterns from other spurious patterns. This is identical to one of the problems faced by this LDRD project – the isolation of important infrequent event patterns that indicate an anomaly in system operation or unauthorized bunker activity. In this work, Karypis and Kumar propose to build on their previous work in temporal patterns [15], which uses a directed acyclic graph to specify allowed relationships among the objects in patterns, facilitating the discovery of arbitrarily complex temporal patterns. They also plan to extend this framework to include spatial predicates to find spatio-temporal patterns.

## 2.2 AVATAR

The goal of the AVATAR project [16] is to capture what the user thinks are interesting patterns in the dataset during the visualization of scientific simulation results, and use this information to automatically point the user to similar interesting data in other regions or in another dataset. The approach can be broken into three primary steps: 1) User-driven pattern discovery through a modified version of the MUSTAFA visualization tool, 2) Learning, and 3) Pattern detection. User-driven pattern discovery is a key concept embodied in the Sequence Pattern Discovery Tool developed in this LDRD project.

### 2.2.1 User-Driven Pattern Discovery

To capture interesting patterns in the data, the MUSTAFA visualization tool was modified to allow the user to draw a rectangle over regions in the current view and label the underlying mesh nodes in these regions as Very Interesting, Interesting, Probably Interesting, or Not Interesting. Unlabeled portions of the data were subsequently labeled as Unseen low or Unknown. The labeled data containing the user-discovered patterns is saved to a file, where each mesh node contains the original fields that hold the physics variables etc., in addition to a new field that holds the label.

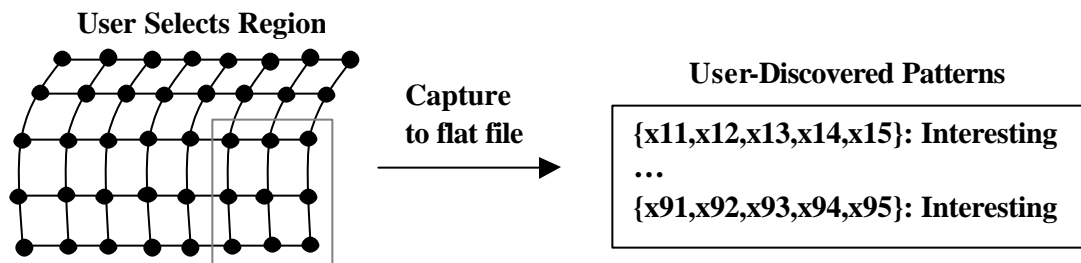


Figure 1: Capturing user-discovered patterns

### **2.2.2 Learning**

The learning process consists of inducing a decision tree on the user-discovered patterns. A parallel approach is needed since the user-discovered patterns can consist of very large amounts of data. The challenge is to build a parallel decision tree inducer that does not decrease in accuracy as the dataset scales in size. The simple, yet effective solution developed was to partition the data into  $N$  disjoint (or overlapping) subsets, and place one subset on each of the  $N$  processors of the parallel machine. Each processor then invokes the C4.5v8 algorithm [17][18] to construct a decision tree independently of the other processors. This eliminates the need for inter-processor communication during tree induction. Each decision tree is saved and distributed to the other  $N-1$  processors for pattern detection.

### **2.2.3 Pattern Detection**

After tree induction, each processor has all  $N$  induced decision trees. To detect patterns in new data (i.e. to classify the data), it is partitioned among the available processors, and each processor runs its portion of the data through all of the decision trees and uses a majority vote of the results to determine the final classification of each piece of data. This simple, scalable process was shown to have similar accuracy to the serial version of the algorithm. A weighted voting scheme was also considered; however, results did not show an improved accuracy over the simpler majority voting scheme. Currently, the AVATAR system runs on ASCI Red and ASCI Blue utilizing MUSTAFA and Exodus datasets. A large example, consisting of 800,000 training examples on each of 64 processing nodes (51,200,200 examples) was completed on ASCI Red in 2.5 hours. The key barrier to scalability of the parallel classifier is the limited parallel I/O capability of ASCI Red.

## **2.3 Feature Characterization in Scientific Datasets**

Like the Pattern Discovery in Scientific Data project, the FCDMF (Feature Characterization using Data Models and Formats) project [19] seeks to develop methods to mine higher-level objects from scientific simulation results. A hierarchical feature ontology is used that contains a base layer of objects that violate basic continuity and smoothness assumptions, and layers of higher-order objects that violate laws of specific domains. This ontology is used to mathematically describe and characterize features automatically, so the analyst can focus on the important regions that require close study. This effort relates to our LDRD project in that it develops a data analysis tool, much like the developed Sequence Pattern Discovery Tool to aid the user in discovering interesting patterns by performing automatic and user-directed computations on the data.

### **2.3.1 Data Format**

When constructing a general-purpose tool, it is important for the tool to be able to read and write data in common formats. The FCDMF project addressed this issue by constructing the tool around the SAF [20] data format. SAF provides for the flexible representation of a wide variety of scientific data by incorporating metadata (or data about data) as part of the format. At any level of abstraction, computed features can be stored along with the raw mesh data as part of the metadata. The SAF libraries provide

for the automatic translation between different types of data to facilitate its use by various different tools.

### 2.3.2 Features as Patterns

The base ontology which forms the core of the feature set used by FCDMF consists of features, or computable patterns in the scientific data, that express the violation of the continuity and smoothness assumptions that are intrinsic to the laws of physics and numerical simulation. These base features include cracks, spikes, tears, wrinkles, etc. Higher levels of the feature ontology, which are constructed from component features in the base level, include features that express the violation of higher-level physical laws, such as the contact problem, where normal forces between two surfaces in contact deviate from being equal and opposite.

This hierarchical pattern relationship is not unlike the construction of larger patterns from subsequences of tokens in the Pattern Discovery Tool. In both cases, the interesting patterns are those that do not occur frequently. For example, the Pattern Discovery Tool may be used to isolate an infrequent token sequence indicating an unauthorized access event, and the FCDMF algorithms can be used to identify single deviations in large sets of scientific data, such as a spike in temperature at a hot point.

### 2.3.3 Framework Algorithms

To compute the various features in the feature ontology, a set of algorithms were developed as feature building blocks:

- Normals(): takes a SAF dataset and computes the unit-length normal vector to each element of the mesh.
- Topological-neighbors(): takes a SAF dataset and an individual mesh element,  $m$ , and returns a list of mesh elements that share an edge or vertex with  $m$ .
- Geometric-neighbors(): takes a SAF dataset, and individual mesh element,  $m$ , and a radius,  $r$ , and returns a list of mesh elements whose vertices are within the Euclidean distance of  $r$  from  $m$ .
- Statistics(): takes a SAF dataset and a specification of one variable (either a mesh coordinate or a physics variable), and computes the maximum, minimum, mean, and standard deviation of its values.
- Displacements(): takes a SAF dataset and finds all topologically neighboring pairs of vertices, measures the  $xyz$  distance between them, and reports the maximum, minimum, mean, and standard deviation of those distances.

In addition to these algorithms, other fundamental vector calculus computations are provided in the FCDMF framework. The Pattern Discovery Tool also contains a set of algorithms to construct, compare, and filter patterns, either automatically, or under user control. These are manifested in the various menus and pop-ups in the user interface.

## 2.4 VxInsight

VxInsight is a knowledge management and visualization tool for discovering relationships within large databases [21]. Rather than invoking standard clustering

algorithms and displaying the data to the user, VxInsight instead computes a similarity metric on the data and displays data points for each datum, where data appear geometrically closer to each other in the visual display as their similarity value increases. The underlying concept is to use the human's exceptional ability to visualize patterns, relationships, trends, and anomalies. Data is displayed as points on a 3D landscape. Mountains in the landscape indicate data that are similar to each other, and the height of the mountain indicates the density of the data in that region. Intuitively, the user can navigate the data landscape much like one would navigate a 3D map of a geographical region. VxInsight [21] claims to improve on similar visualization tools, such as SGI's MineSet [22], by providing dynamic peak labels to provide navigational guidance through the data landscape. The Pattern Discovery Tool is similar to VxInsight in the context of providing user-directed pattern discovery.

VxInsight allows the user to navigate through the data and visually identify and explore patterns in the data through color or topology, while the Pattern Discovery Tool developed in this LDRD presents a preliminary set of patterns to the user which can be filtered, merged, and extended through interaction with the tool to isolate interesting patterns in a user-determined slice of the data.

#### **2.4.1 Computing Object Similarity**

The specific function used to determine similarity between data objects in VxInsight is dependent upon the domain of the data. For example, the metric could be based on common keywords in documents, identical vocabulary within documents, citation links between scientific papers or patents, direct links in web documents, financial transaction links between corporations, or membership in common organizations among individuals [21]. The general function maps object pairs to non-negative real numbers. The greater the similarity between the two objects, the larger the number returned by the similarity function.

Similarly, patterns identified in the Pattern Discovery Tool are compared based on the similarity of the representative token sequence string. Three component metrics are used to compute the similarity value of the two strings based on relative character position, the number of missing characters, and the difference in length of the strings. These metrics are normalized and combined to generate an overall similarity metric whose range is from 0.0 to 1.0. The higher the normalized value, the more similar the strings, and therefore, the more similar the token sequence patterns. So both VxInsight and the Pattern Discovery Tool quantify similarity by mapping it to non-negative real numbers.

#### **2.4.2 Navigating Science**

The primary application and the chief motivation for the development of VxInsight is to determine what scientific efforts to support with research funding to yield the greatest impact [22][23]. To help do this, VxInsight was used on a database of scientific papers where initial similarity was based on citation content. VxInsight was used to determine where a particular body of work originated, how it has evolved in the past, and the trends for future research. VxInsight has also been applied to study nuclear proliferation [24]. The structure of nuclear technology literature was first visualized, then used with text

analysis tools to determine similarities between papers and public sources discussing nuclear technologies. Analysts were then able to track down potentially sensitive information using VxInsight as a form of intelligence gathering tool. Other applications of VxInsight include: 1) exploring citation indices to identify similar efforts in the laboratories and in industry and point to areas of possible collaboration, 2) detecting Medicaid fraud, 3) counter-terrorism intelligence, and 4) characterizing the nature of patent citations.

### **2.4.3 Moving Forward**

From the above discussion of related work at three DOE laboratories in the NWC, we note that the work in this project is similar to and applicable to many of the related works, but yet is distinct in the problems addressed in pattern discovery. In particular, we address the problem of how to automatically discover patterns in event sequence data.

In the next section, we discuss our initial investigation of two similarity mechanisms, Dynamic Time Warping, and Hidden Markov Models, which were explored early in this project as possible solution approaches.



## 3 Dynamic Time Warping and Hidden Markov Models

As noted in the introduction, the test data we had was a mixture of measurements of continuous-valued quantities and discrete events. In order to automate analysis of this data, the two types of data must first be placed on an equal footing. The approach taken in the Knowledge Generation software is to extract key features from the measurements of continuous-valued quantities that can be mapped to significant events in the monitored processes. The Knowledge Generation software suite includes a rule editor that allows a user to define a set of rules for extracting events from the continuous-valued data.

However, our experience with test data showed that it can be difficult to write a robust set of rules for feature extraction. In some applications, the time allowed certain steps in a monitored process varied greatly. In addition, radiation level measurements were sensitive to geometry as well as history of the radioactive material. Our motivation for exploring the pattern discovery and matching problem in the continuous-valued data was to develop algorithms to aid in writing robust feature extraction rules.

A literature search on pattern matching in continuous-valued data turned up significant features in the problem we were trying to address. Whereas the available literature concentrated on handling large quantities of evenly spaced data, we had sparse, irregular data. Furthermore, as described above, our pattern-matching problem is complicated by the fact that we would like to allow the patterns to be able to stretch in both the time dimension and in the dependent variable dimension. In addition to pattern matching, we also aimed to discover new, yet unknown patterns in time series data. Two approaches for measuring similarity were identified as likely candidates: Dynamic Time Warping (DTW) and Hidden Markov Models. The former looked particularly interesting at the outset because some of the papers described segmenting the data in order to reduce computation time on large data sets. Segmentation of the data could be a step toward feature extraction.

Both algorithms are described in detail below, along with their respective advantages and disadvantages. Although the DTW approach looked most promising, in the end, we had insufficient data to make much progress on this thrust.

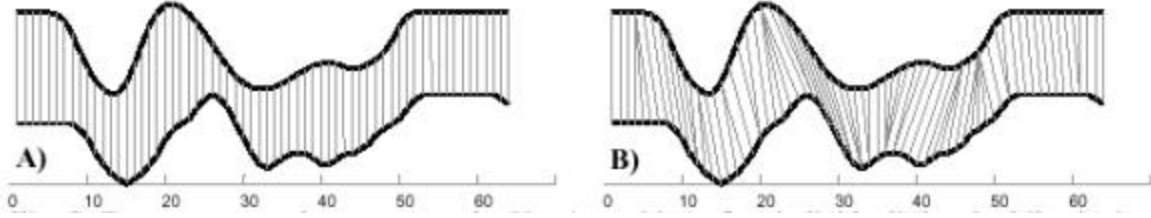
### 3.1 *Dynamic Time Warping*

DTW is defined in [10] by the following:

Given two time series,  $O_1$  and  $O_2$ , DTW finds the warping of the time dimension in  $O_1$  that minimizes the difference between the two series.

In essence, DTW is a technique for aligning time series along the time axis for a better similarity measure (see Figure 2). As humans, we can see that the two sequences have an overall similar shape, but they are not aligned along the time axis. In Figure 2(A), the similarity measure between the two sequences is based on Euclidean measure, which assumes that the  $i^{th}$  point in one sequence aligns with the  $i^{th}$  point in the other sequence.

This produces a very pessimistic similarity measure. On the other hand, DTW aligns the sequences in such a way as to minimize the distance between them. This nonlinear alignment results in a more sophisticated distance measure as shown in Figure 2(B).



**Figure 2: Comparison of similarity measure based on Euclidean distance and DTW. (A) A similarity measure based on Euclidean distance (the  $i^{th}$  point of one sequence is aligned with the  $i^{th}$  point of the other sequence) produces a pessimistic similarity measure. (B) DTW produces a better similarity measure. Figure taken from [9].**

There are numerous papers on using DTW for time series data, including [6][7][8][9][10]. Berndt and Clifford [6] describe a DTW algorithm as follows:

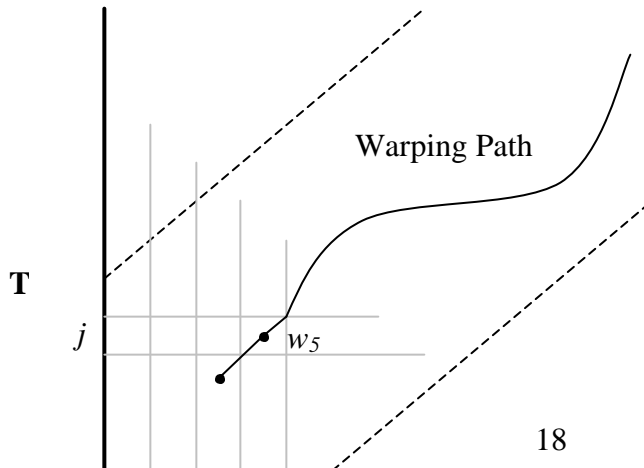
Assume we have two time series,  $S$  and  $T$ , where:

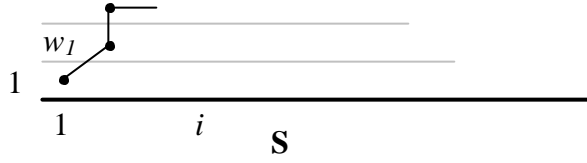
$$S = s_1, s_2, \dots, s_i, \dots, s_n$$

$$T = t_1, t_2, \dots, t_j, \dots, t_m$$

Note that the elements of the series are numbered according to where they occur in the series, not according to the time at which they occur. Thus,  $s_i$  is the  $i^{th}$  element of the series. The starting points for each series ( $s_0$  and  $t_0$ ) are given to the algorithm. Hence,  $S$  and  $T$  need not be aligned in the time axis, which allows for comparison of sequences that do not start at the same time.

The first step is to arrange the sequences  $S$  and  $T$  to form an  $n$ -by- $m$  grid, where each grid point,  $(i, j)$ , corresponds to an alignment between elements  $s_i$  and  $t_j$ . We will use this grid to find the warping path,  $W$ , which aligns the elements of  $S$  and  $T$  such that the distance between them is minimized. Let  $W = w_1, w_2, \dots, w_p$ , then the warping path is a sequence of grid points where  $w_k = (i, j)_k$ . Figure 3 shows an example of a warping path where the point  $w_5$  indicates that  $s_i$  is aligned with  $t_j$ . Naturally, if there is no difference in the time axis between two time series, then  $W$  corresponds to points along the line  $i = j$ .



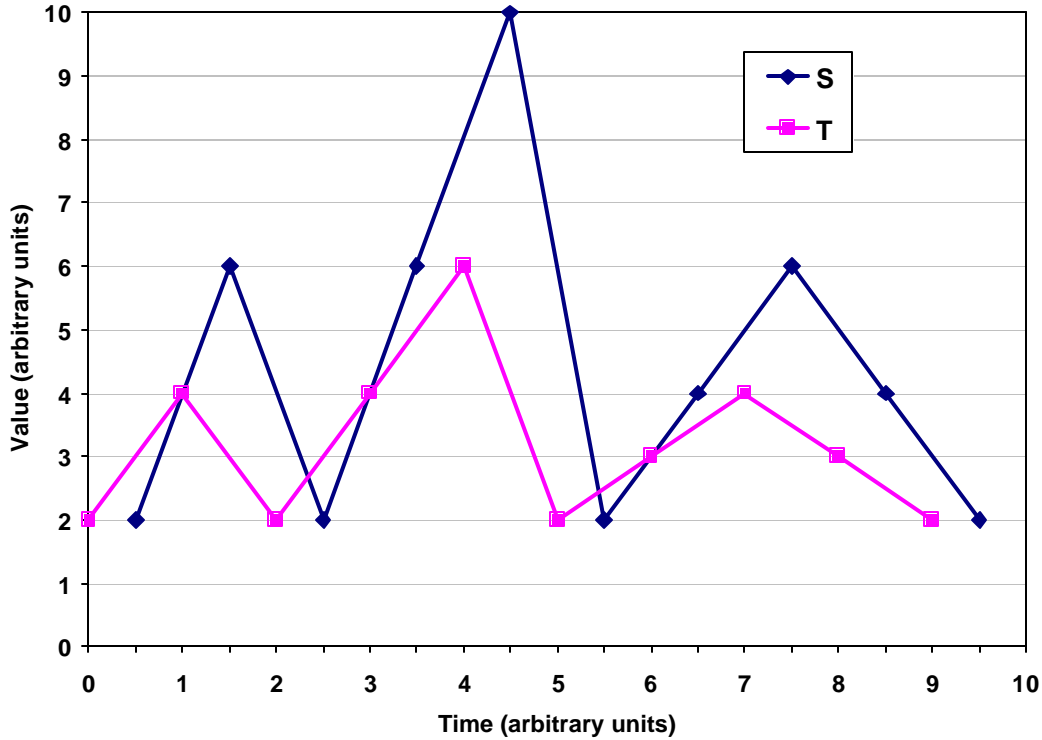


**Figure 3: An example of a warping path.**

For detailed discussions regarding distance measures, normalization, and efficiency restrictions (including boundary conditions, warping windows, and other constraints), the interested reader is referred to [6]. Below we present an example which utilizes one instantiation of Berndt and Clifford's DTW algorithm.

### 3.1.1 An Example of DTW

Consider the two time series,  $S$  and  $T$ , as shown in Figure 4. The two series are similar. Note that  $S$  is offset from  $T$  along the time axis, and  $S$  is stretched vertically to be twice  $T$ . The values along the y-axis have no particular meaning in this example.



**Figure 4: Example time series.**

From Berndt & Clifford's algorithm, we establish an  $n$ -by- $m$  grid with  $S$  on one axis and  $T$  on the other, as shown in Figure 5. The number contained in each matrix element  $(i,j)$  represents the cumulative minimum distance between the  $i^{th}$  point of  $S$  and the  $j^{th}$  point of  $T$ . This is calculated using the following equation:

$$D(i,j) = d(i,j) + \min[ D(i-1,j), D(i-1,j-1), D(i,j-1) ]$$

where  $d(i,j)$  is the distance between  $i$  and  $j$  based on some distance measure. Essentially, this equation is stating that the cumulative distance,  $?(i,j)$ , is the sum of the distance between  $i$  and  $j$  (specified by a point) and the minimum of the cumulative distances of all neighboring points. In this example, the magnitude of the difference between the values is used as the distance measure, i.e.,  $d(i,j) = |s_i - t_j|$ . For example, the calculation for the grid location  $(5,5)$ , which calculates the cumulative distance between  $s_5$  and  $t_5$  follows:

$$\begin{aligned} ?(5,5) &= d(5,5) + \min[?(4,5), ?(4,4), ?(5,4)] \\ &= |10 - 6| + \min[4, 4, 10] \\ &= 4 + 4 \\ &= 8 \end{aligned}$$

Upon completion of all calculations, the optimal warping path is found by backtracking through the grid and selecting the previous points with the lowest cumulative distance, as highlighted in Figure 5. As you can see, DTW successfully generates what can be

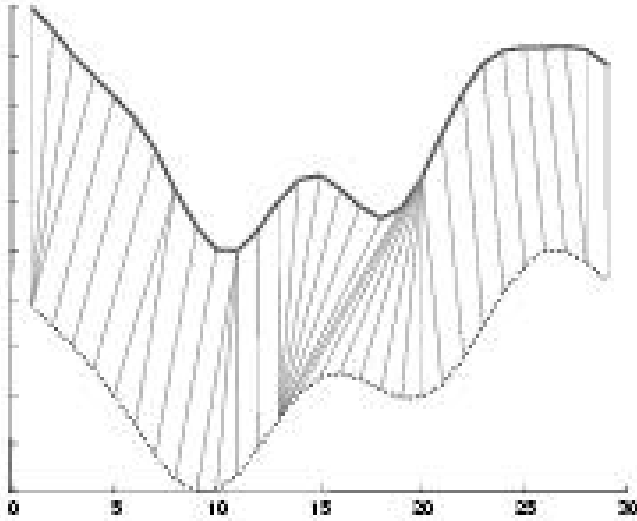
T	10	12	16	8	12	18	12	12	14	14	12
	9	12	14	8	10	14	12	10	12	12	12
	8	11	11	7	7	13	11	9	11	11	13
	7	9	11	5	7	14	9	9	12	13	14
	6	8	8	4	8	12	8	10	14	14	14
	5	8	4	8	4	8	12	14	12	14	18
	4	4	4	4	4	10	12	12	14	14	16
	3	2	6	2	6	14	12	14	18	18	16
	2	2	2	4	6	12	14	14	16	16	18
	1	0	4	4	8	16	16	18	22	24	24
											S
											1
											2
											3
											4
											5
											6
											7
											8
											9
											10

intuitively seen as the “correct” points of alignment.

**Figure 5: The  $n$ -by- $m$  matrix with the warping path highlighted.**

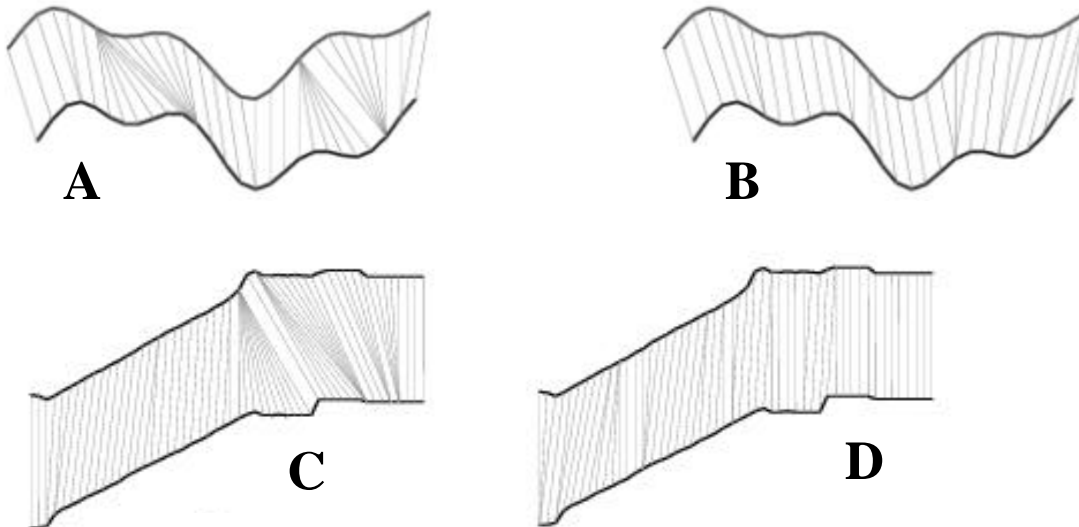
### 3.1.2 Problems with DTW

DTW has been successfully used in many domains, and it looked like a very promising way to compare a known pattern to a current data set in order to find if and how often it occurred. However, DTW can produce unintuitive alignments where a single point in one time series is mapped onto a large subsection of the other time series. In other instances, DTW fails to find obvious alignments because a peak, valley, plateau, or other feature of one series is slightly higher or lower than the corresponding feature in the other series (see Figure 6).



**Figure 6: Example where DTW does not find the obvious, natural alignment. Note that it failed to align the two central peaks. *Figure taken from [9].***

Extensive discussion with Eamonn Keogh led to his idea is to perform DTW on the derivative of the time series instead of on the time series itself. This is termed Derivative Dynamic Time Warping (DDTW). Figure 7 compares the results of DTW and DDTW for time series comparisons on which DTW fails to find the optimal alignment. (A) and (C) show examples of alignments produced by DTW on a pair of time series, while (B) and (D) show examples of better alignments produced by DDTW on the same pairs of time series.



**Figure 7: Comparison of DTW and DDTW. (A) and (C) show problematic alignments produced by DTW. (B) and (D) are alignments produced by DDTW. *Figures taken from [9].***

Despite the promise of this approach, time constraints prohibited the opportunity to explore the DDTW algorithm further on our data sets.

## 3.2 Hidden Markov Models

We briefly spent some time exploring Hidden Markov Models (HMMs) as another approach to solve our problem. The following is only a brief overview of HMMs, but the interested reader is directed to a comprehensive tutorial about HMMs in [25].

An HMM consists of the following items:

- ?? A set of  $N$  states,  $\{1, \dots, N\}$
- ?? An alphabet of output symbols,  $\{a, \dots, z\}$
- ?? A set of output probabilities, the probability that a particular symbol will be emitted while in a given state.
- ?? A set of transition probabilities, the probability that the model will transition (or jump) from the given state to any other state at the next time step.
- ?? An initial state probability distribution.

At each time step, the HMM emits a symbol and either stays in the same state, or transitions to another state.

Using this type of model, one can ask questions such as: What is the probability that a given HMM generated the sequence  $a,b,a,c$ ? Several transitions could have produced that sequence. The Viterbi algorithm works out an approximation to the probability that a particular sequence of observations was generated by an HMM. They are called *Hidden Markov Models* because the states that the system goes through are not known.

### 3.2.1 Advantages and Disadvantages of HMMs

The advantage of HMMs is that they are firmly grounded in probability. There are principled and systematic procedures for estimating and training the model from labeled data as well as standard methods for detection, such as the Viterbi algorithm. When the Viterbi algorithm is implemented, the resulting number has meaning - it is the probability that a time series was generated by an HMM along the most probable path through the HMM for that sequence. Conversely, the number that comes out of the DTW algorithm is not as explainable, and its interpretation as a similarity measure is series-dependent.

The disadvantage of HMMs is the same as the advantage: they are firmly grounded in probability. HMMs are much more difficult to understand and implement. Show a person a picture of two time series stretching and aligning along the time axis as in Figure 2, and they will understand it. Show that same person a page of probability figures and state transition diagrams for the corresponding HMM, and they will most likely not understand it. In addition, it has been proven in [10] that HMMs and DTW are, in fact, optimizing the same criterion. In his communication with the team, Tim Oates suggested that “if DTW does the job for you, the algorithm is much easier to implement and might be computationally cheaper than the corresponding HMM algorithms.” This led us to pursue DTW in greater depth than HMMs.

### **3.3 Directions for Further Work**

Our initial investigation of Keogh's DDTW algorithm indicates that this is a fruitful approach for further study. Another interesting approach is that of Oates, Firoiu, and Cohen, who introduce and discuss the idea of combining DTW and HMMs in [10]. This work is expanded further in Oates' dissertation, titled *Grounding Knowledge in Sensors: Unsupervised Learning for Language and Planning*.

The next section discusses our approach and solutions to the pattern discovery problems we have addressed, in particular, pattern discovery in discrete event time-series data.

## 4 Pattern Discovery in Discrete Event Time Series Data

### 4.1 Introduction

The second major emphasis of the LDRD project was the problem of discovering patterns in discrete event time series data. We conservatively define a pattern as a subsequence that occurs more than once, although a more general definition for a pattern is subsequences that occur more frequently than some threshold. The approach developed in this section is equally applicable to discovering patterns in both event data from a single sensor and event data from many sensors.

The basic approach is to tokenize the event data, representing the events as symbols in a sequence. In some cases, it is possible to use domain-specific knowledge to pre-process the sequence and reduce its size. Initially, patterns are “discovered” in the data by enumerating all subsequences that occur more than once. We generally constrain the size of the subsequences to some user-defined value. In addition to frequency, other statistical measures can be computed to help a user decide which patterns are significant.

One interesting problem with discrete event data from multiple sensors is that events from simultaneous activities can be intermingled, effectively disrupting the patterns or making them “noisy.” To handle this problem, we investigated the use of regular expressions and likeness measures to identify similar patterns.

### 4.2 Event Tokenization

The process of tokenization is straightforward and is dependent on the specific data. The raw data will be in a record form containing time and event information including the source of the event and the event type (e.g., door open, door close, break-beam broken, etc.).

There are a number of different tokenization schemes that could be applied to the data. The simplest scheme we have used involves time flattening. In a time-flattened sequence we tokenize strictly on the event type in chronological order, ignoring both the duration times between events and the source of the data. Table 1 shows a portion of a simple time-flattened, event-type/token mapping. A portion of a tokenized sequence is shown below:

ADED FDEFDEFDEFDEFDEFFDEDFDE ...

**Table 1: Bunker sensor event type tokens**

Event Description	Token
Door Open	A
Door Close	B
Motion Start	C
Occupied	D



Image Trigger	E
Unoccupied	F

The monitoring system that provided much of the data studied during this project has a number of sensors that behave similarly and the observed activities resulted in considerable variability in relative timing of sensor events. The result of this simple time-flattened tokenization approach was quite powerful in detecting generic patterns.

More complicated tokenization schemes can be used to capture other dimensions of the event data. For example, the time duration between events may be important in deciding if two event sequences are similar. To capture this aspect of the problem, we experimented with a tokenization scheme in which the event types were tokenized with capital letters as shown above. The time between events was then represented with the lower case letters "a" through "j" where "a" represented 0 to 6 minutes since the previous event, and "b" 6 to 12 minutes and so forth. The letter "j" represented an hour or more between events. Using the same data as the previous example, the resulting sequence is shown below:

AaDaEaDaFaDaEaFaDaEaFaDaEaFaFaDaEaDaFaFaDaE ...

The tokenization scheme can also be extended to handle complex objects. In fact, this is the approach used in the robot data analysis example in Section 5.8.3

For the remainder of this report, the time-flattened token sequences will be used in order to keep the examples simple.

### 4.3 Preprocessing the Sequence

There are two ways the sequence may be pre-processed before beginning the pattern discovery process. First, repeated tokens can be condensed into a single token, producing a form of event collapsing. This is useful in cases in which certain event types repeat frequently (e.g., motion events from a motion sensor) and the user has decided that the number of repeated events is of little interest in identifying patterns. The token condensation process probably makes the most sense for time-flattened sequences.

In addition to generating single symbol sequences, pre-processing can also generate "Event Words" which is the segmentation of the sequence based on a specified arbitrary time interval between event tokens. This approach is useful if the data captures several periods of sporadic activity. A relatively long time between events indicates a quiescent state for the monitored facility. By dividing the data into event words, similarities among the periods of activity can be identified. The following sequences show examples of both condensed sequences (1) and event words with a maximum interval of 12 minutes (2):

(1) ADEDFDEFDEFDEFDEFDEDFDE

(2) JEJEJEJEJ

ADECFCDEDCEFFDEDCFBCEF  
J  
J  
J  
ADEDIFCDECFCDECBCECECEFF  
CECECEC  
CE  
CECE  
CECECECECE

## 4.4 Pattern Discovery

The primary purpose behind our efforts is to discover or mine for new and interesting patterns in the data. The method we pursued to find new patterns was to exhaustively search the symbol sequences for repeating subsequences. This method will produce volumes of subsequences depending on the length of the sequence and the distinct number of tokens (alphabet) that comprises the sequence. There are two problems associated with pattern discovery. The first problem is the intensive nature of the exhaustive search required to find all patterns. The search is exponential to the size of the sequence. Second is the matter of deciding what subsequences or patterns are significant and merit further investigation or should be ignored.

One of the data sets studied during the course of investigation in the LDRD was the Bunker Sensor data. This data set is essentially a list of events from various sensors located in storage bunkers. These events are recorded in a database as the events occur. These events are tokenized as symbols in a time-flattened sequence, which represents the list of events in chronological order. The goal of our study was to discover patterns that occur in the event list. Further, the significant patterns may not be exact matches but reasonable approximations. The first step in our process is to discover patterns that are exact matches and are frequent. Our initial attempts employed a brute force method that enumerated every possible subsequence of every possible size. Once enumerated the list can be pruned to eliminate non-frequent patterns and patterns that are wholly contained in larger subsequences. This brute force approach is certainly capable of achieving the goal. It is however very intensive in both space and time requirements. The number of subsequences that must be examined is  $N(N-1)/2$  where  $N$  is the length of the overall sequence. For a sequence with 1000 elements, 499,500 subsequences must be examined. It is reasonable to assume that the unsupervised search for unknown patterns will be intensive; however, there is a possibility of reducing the search space using pruning techniques.

### 4.4.1 Related Efforts

Agrawal and Srikant at the IBM Almaden Research Center introduced two important data mining algorithms known as A-priori [11] and GSP (Generalized Sequential Pattern mining) [30]. The basic concepts underlying these algorithms are as follows. If a subsequence of length  $k$  is determined to be frequent in a sequence and  $k$  is greater than 1, then any subsequence of length  $k - 1$  from the previous subsequence must also be

frequent. A subsequence is deemed to be frequent if the count of occurrences of that subsequence in a larger sequence is greater than some predefined threshold. The discovery of new frequent patterns is done in a stepwise progression where new patterns of size  $k$  are found by building upon patterns of size  $k - 1$ . A-priori and GSP build the next level of new patterns by generating the set of possible next level sequences then scanning the database for their existence.

#### 4.4.2 Adaptations to A-priori and GSP

We have developed an algorithm that incorporates the incremental creation of patterns using the frequency-pruning model from A-priori and GSP. This adaptation views the sequence as an indexed array and the subsequences are simply represented as an index pointer and a sequence length. Rather than generating all possible next level sequences and scanning for their existence, this algorithm takes the previous patterns of length  $k - 1$  and generates the next set of subsequences with length  $k$ . The distinct subsequences are tallied to determine frequency by maintaining a hash table of the subsequences and their counts. For each new distinct pattern generated, the frequency is checked and those not meeting the frequency threshold are pruned from the list of new patterns. After the list of new patterns is pruned the remaining patterns are added to the list of frequent patterns.

#### 4.4.3 Pattern Significance

Our algorithm takes the pruning phase one step further by eliminating any patterns of length  $k - 1$  from the list of frequent patterns if those patterns are wholly contained in the patterns of length  $k$ . We accomplish this by comparing the frequency of a pattern with the frequency of the prefix of the pattern. If the frequency of the prefix matches the frequency of the pattern, then the prefix is wholly contained in the pattern and the prefix can be eliminated from the list of frequent patterns. This is also true for the suffix of the pattern. Consider the following example.

**Table 2: Patterns for sequence ABXABZ**

Frequent Pattern	Sequence Positions	Frequency
A	0, 3	2
B	1, 4	2
AB	0, 3	2

The pattern AB starts with the token A, and since the frequency of both patterns (A and AB) is 2, it is clear that each occurrence of A is found in every occurrence of AB. A is wholly contained in AB, and in our algorithm, the prefix A can be eliminated from the list of frequent patterns. Note that the suffix B is also wholly contained in the pattern AB in this example and can also be eliminated. Had the example sequence been ABXABZA then the frequency of the token A would be 3 and A would not be eliminated.

#### **4.4.4 Java Implementation**

While pursuing our investigation we built a Java implementation of this algorithm and incorporated it into the Pattern Discovery Tool. Java's object orientation facilitates a highly flexible implementation. The fixed array was implemented as an array of objects, which means that the array elements can be a sequence of any object that represent an entry in the data set as needed without requiring a modification to the algorithm. The keys used for the hash table entries are simply array pointer objects that point to the first occurrence of a discovered pattern sequence in the array. The only requirements imposed on any new application object is that it must implement a Java class that represents a single entry in the data set, and provide methods to assess equivalence of the entries.

#### **4.4.5 Future Investigation in Sequence Pattern Discovery**

Much of our investigation centered on sequence mining; however, the temporal aspects of event sequences should also be considered. The study of time granularities and temporal reasoning [12] in conjunction with pattern discovery will enhance our ability to understand time sequences where events are not distributed in periodic patterns.

There are a number of other individuals working on similar problems that came to light late in the LDRD cycle. Of particular interest are a number of papers presented at the Knowledge Discovery and Data mining conference in August 2001. Mannila and Salmenkivi [27] provided a study of event intensity in sequences. Liu, Hsu and Ma [28] propose a methodology to remove insignificant rules or sequences. Zheng, Kohavi and Mason [29] provided a well-researched comparative analysis of the leading association-rule discovery algorithms. Han and Pei [30] presented two A-priori algorithm adaptations, the FreeSpan and PrefixSpan algorithms. These algorithms are logical performance extensions to A-priori and GSP.

### **4.5 Post Processing and Analysis**

After the initial discovery of patterns, additional processing and analysis can take place to mine interesting information from the set of discovered frequent patterns.

#### **4.5.1 Elementary Patterns**

The premise that states that any pattern that is wholly contained within larger patterns is insignificant may or may not prove valid. The premise does prove useful in reducing the number of patterns to consider. However, the premise ignores the possibility that there are smaller or more elementary patterns that are building blocks of larger patterns. Running the pattern detection process and the reduction algorithms on a data set of approximately 2000 bunker events produced more than 350 patterns. Table 3 shows a sample of these patterns that contain a possible elementary pattern CE. This elementary pattern by itself was removed during the pattern discovery process, as were CECE, CECECE, CECECECE and CECECECECE. It appears that CE by itself has some significance because there are patterns where it occurs without repetition, but in conjunction with other tokens. It also appears however, that the patterns where CE repeats, such as CECE, are probably not significant by themselves. More work needs to be done to find a solution for this issue.

**Table 3: A sample of patterns containing the possible elementary pattern CE**

Pattern	Size	Frequency
ADEDCEC	7	2
BCECE	5	3
BCEF	4	3
CBCECEC	7	2
CDECE	5	2
CEAC	4	2
CECECDEC	8	2
CECECEC	7	219
CECECECEC	9	207
CECECECECECE	12	198
CECECEO	7	2
CECECF	6	6

#### 4.5.2 Noise Avoidance

Event collapsing techniques are a form of noise avoidance in that they remove uninteresting subsequences from consideration, but they only suppress events of a specific type. There is a need to find patterns that may have arbitrary "noise" or unrelated events included. Certainly, a studied approach to the analysis of the data and the removal of uninteresting events could reduce much of the noise. Such an approach may be feasible, but would be costly because of its need for human interaction. The need to detect similar patterns provides opportunities for future work. Some possible techniques include Regular Expressions [31] and other State Machine models especially if they incorporate the temporal aspects of the data.

As an example, the following are a set of regular expressions that can be generated from a selected subsequence (pattern). These regular expressions can then be executed against the larger sequence to find other subsequences that match them. These new subsequences should be reasonably similar to the source pattern, but containing some noise. These expression derivations assume that the positions of the first and last tokens are significant.

Two character pattern - i.e. AB:

Boundary Expression - "A.\*?B" - find any sequence that starts with an A and ends with a B and has anything else in between.

Three character pattern - i.e. ABC:

Boundary Expression - "A.\*?C"

Interior-Permutation-with-Noise - "A.\*?B.\*?C" - find any sequence that starts with an A and ends with a C and has at least one B in between.

Four or more character pattern - i.e. ABCD:

Boundary Expression - "A.\*?D"

Interior-Permutation-with-Noise Expression -

"A.\*?B.\*?D", "A.\*?C.\*?D", "A.\*?B.\*?C.\*?D", and "A.\*?C.\*?B.\*?D"

### 4.5.3 Statistical Measures

Statistical analysis is the most promising tool for determining the significance of a discovered pattern. The designations that we use require some explanation. MF stands for measured frequency and MP for measured probability. MF is simply the count of occurrences of a given pattern in the sequence. MP is calculated as number of occurrences of a subsequence divided by the number of windows in the sequence where the subsequence could occur. We count windows as the number of possible positions in the sequence in which a given pattern could exist. For example if a sequence has 10 positions and the target pattern is 5 tokens long then the number of windows is 6. The number of windows is calculated as  $\text{Sequence\_Size} - \text{Pattern\_Size} + 1$ . The letters A, B and C represent individual tokens found in the pattern regardless of the actual number of actual tokens. If the letters are shown individually, the implication is that the individual tokens in the pattern are considered individually. If the letters are contiguous then the tokens are used as a group in the calculation of the statistic. Each measure that we have explored will be described below.

MP(A)MP(B)MP(C)

This statistic is the measured probability of each individual token in the pattern multiplied by the measured probability of the other tokens in the pattern. It answers the question, if tokens are occurring randomly, what is the probability that they will occur together at any given window. In our data the  $\text{MP}("D") = 0.103071$  as "D" occurred 198 times within 1921 windows of size 1, and the  $\text{MP}("E") = 0.281624$  as "E" occurred 541 times, therefore the  $\text{MP}("D")\text{MP}("E") = 0.029027$

MF(MP(A)MP(B)MP(C))

This statistic measures the expected frequency of a set of tokens occurring together in the sequence, assuming random placement of the tokens. It is calculated by multiplying the  $\text{MP}("A")\text{MP}("B")\text{MP}("C")$  by the number of windows for the size of the given pattern. In our example  $\text{MF}(\text{MP}("D")\text{MP}("E")) = 55.73184$  or we would expect to find the pattern around 56 times if D and E are placed randomly in the sequence.

MP(ABC)

This is the probability of a given pattern of any size occurring in any single window. It is calculated as the number of occurrences divided by the number of possible windows. In our example DE actually occurred 144 time in possible 1920 windows, thus  $\text{MP}("DE") = 0.075$

MF(ABC) or Count

This is simply the number of occurrences of a given pattern in the sequence.

MP(A)MP(BC)

This statistic is most meaningful for patterns with more than two tokens. It provides the probability that the first token in the pattern is randomly placed with the remainder of the tokens. When compared with  $MP(ABC)$  it will provide a measure of correlation between the tokens in the pattern.

$MP(AB)MP(C)$

This statistic is used the same as  $MP(A)MP(BC)$  except it operates on the last token rather than the first.

We also investigated the purely random probabilities and frequencies,  $RP(ABC)$ ,  $RF(ABC)$ ,  $RP(A)$  and  $RF(A)$ . These metrics assume that all tokens are equally likely in the sequence. Therefore, these statistics may not be as useful as the measured probabilities and frequencies; they are highly dependant on the length of the sequence and the subsequence.

These statistical measures can be used to answer specific questions about characteristics of the token sequence and the domain it encodes. As an example, consider the following question: Given the token pair AC in the current sequence, what is the likelihood that the token E comes next, i.e. ACE? In the bunker domain from Table 1, this question translates to: Given a Door Open event followed by a Motion Start event, what is the likelihood that the next event is an Image Trigger? To answer this question, we divide the measured frequency of the ACE pattern by the measured frequency of the AC pattern:  $MF(ACE) / MF(AC)$ . If ACE occurred 21 times and AC occurred 32 times, then we could conclude that, in the scope of the current data set, any Image Trigger event immediately followed the Door Open – Motion Start event sequence occurs roughly 65% of the time.

## **4.6 String Likeness Measures**

In the course of investigation of the problem of discovering patterns within symbol sequences, it has become desirable to discover patterns that are like others. Tools such as Regular Expressions can easily enumerate a list of patterns that are similar within a sequence or string of character tokens. Once the list has been compiled it is desirable to determine their likeness as compared to another string, perhaps a pattern that for some reason is considered to be significant. Below we discuss several likeness or fuzzy metrics that can be applied to this problem.

### **4.6.1 Human Interpretation of Likeness**

Likeness or similarity between two character strings is highly ambiguous and subject to individual interpretation. A brief poll was taken, and from 13 responses the foregoing conclusion was drawn confirming intuition. The poll presented 20 pairs of uppercase character strings and asked the respondent to decide how similar the two strings were on a scale from 1 to 10, with 1 being the least similar and 10 being very much the same. No further direction was given as to what the scale meant, and the individuals were forced to decide what string likeness meant to them. The actual poll is shown in the Appendix.

Despite the small sample size (13 responses) from the poll, several trends became apparent in the results. The first string pair ABCD and DEFG had responses that were fairly polar, either hardly alike or very similar. Those feeling that the pair was dissimilar indicated that any similarity based on size and a single common character was rather small. Those stating that the strings were similar did so because they observed a shared lexical sequencing pattern and that the pattern was significant enough to overshadow other dissimilarities. For this pair, responses ranged from 1 to 10 with 7 being the most frequent.

The overall responses tended to exhibit inconsistency in judgment of similarity. In two cases the string ABCDE was paired with the same string, but with interspersed characters (noise). In one case the noise consisted of the single character X, AXBXCXDXEX; the other case used single random characters, AWBOCZDPE. People tended to favor the interspersed Xs over random characters as being more similar, even though there was one more noise-character, X, at the end. It could be concluded that consistent noise is far less distracting than random noise when humans compare strings.

Further analysis is better left to a psychological study. Certainly the poll was an extremely small sample, but interesting conclusions can still be drawn from the data. The purpose of the poll was to gain a rudimentary understanding of how humans interpret string similarity as compared to algorithmic methods. Our goal is not to mimic human behavior but rather to define a metric that can be interpreted usefully by humans when making decisions about the significance of patterns in a data set.

#### **4.6.2 A World of Choices**

If our goal is to provide a metric of comparison between discovered pattern sequences, what are our options? We could try to understand and mimic human behavior. For our specific need this scarcely seems cost effective.

**Basic Metric:** During the course of the study a basic metric was developed. The metric is preliminary, but it provided a starting point for investigation. The metric is a number between 0 and 100 where 0 represents no similarity and 100 represents a perfect match. The algorithm compares two strings by comparing the second string (target string) to the first (source string). Penalties are assessed based on three criteria: 1) the difference in length between the two strings, 2) position discrepancies in the sequence of characters, and 3) missing characters in the target string. If at any time while calculating the penalties the metric falls below zero, then zero is returned.

#### **4.6.3 Related Work in String Similarity Assessment**

The following are a list of methods found from a literature search. Jun-Ichi Aoe [32] describes a number of string comparison metrics. The edit-distance metric measures the smallest number of editing transformations required to transform one string to another. The largest-common-subsequence algorithm is a variation of edit-differences. It seeks to discover the largest subsequence in common between the two strings, and then calculates



the edit differences for the rest of the characters. The k-mismatches metric is a comparison of characters between two strings of the same length. A tally is made of the number of positions between the two strings where the characters do not match. The k-differences metric also calculates the number of edits required for transforming one string into another. The distinguishing characteristic between k-differences and edit-distance is that k-differences considers an update of a character as a single transformation, whereas edit-distance would treat an update as being two transformations, the deletion of one character and an addition of another.

There is an interesting adaptation of the k-mismatches model by Jared Boehm [33]. Each matching character is worth 1.0, if the character is found either immediately before or after its expected location then its worth is discounted to 0.75. This model also allows for phonetic spelling similarity. Exact character matches are worth 1.0, but if a phonetic equivalent were found then the match would be less than 1.0.

#### 4.6.4 Adaptations for Our Needs

The metrics edit-distance, k-mismatches and k-differences do not provide explicit measures for positional or sequencing differences and the interjection of noise; therefore, we decided to explore a refinement of the Basic-Metric described above. Noise may or may not be acceptable depending on what the noise is. Tokens representing system-state-of-health events that are interspersed with tokens representing human-activity events may not be of significance when analyzing events caused by humans. Clearly the absence of some events over others may be significant. Finally, the ordering of tokens may or may not be of consequence depending on the tokens.

We propose a Token Sequence Weighted metric that allows a domain expert to provide appropriate significance to tokens and metric components. Each token can be weighted to provide a penalty relative to the three characteristics, noise, absence, and sequence position. Additionally, the individual components of the metric, noise, absence and sequence position can be weighted to emphasize or de-emphasize their relative importance during analysis.

Noise is a measure of the number of extra tokens found in the target sequence, which are not found in the source sequence. Absence measures the number of tokens found in the source sequence, which are missing in the target sequence. The noise and absence components are calculated by first counting the number of distinct tokens in the source and target sequences. For example, if we wish to compare the token sequence BBAD (target) to ABCA (source) then the following table will be produced:

**Table 4: Token noise and absence analysis**

Token	Source Count	Target Count	Difference $d_i$
A	2	1	1
B	1	2	-1

C	1	0	1
D	0	1	-1

All weights, either for individual tokens or for each of the three components are bounded within the range of 0.0 to 1.0.

**Table 5: Penalty component weighting by token**

Token	Noise Weight $n_i$	Missing Weight $m_i$	Position Weight $p_i$
A	1.0	1.0	1.0
B	0.75	1.0	1.0
C	1.0	0.0	1.0
D	0.5	1.0	1.0

- I.** Noisy tokens appear in the table as negative differences, and absent tokens appear as positive differences. Noise can be calculated by the following formula where  $a$  is the number of distinct tokens in the source and target sequences,  $d_i$  is the difference between the source and target count,  $n_i$  is the token noise weight, and  $t$  is the number of tokens in the target sequence. Note that dividing the summation by  $t$  normalizes this component to a range between 0.0 and 1.0.

$$\frac{1}{t} \sum_{i=1}^a D_i n_i, t \geq 0 \quad \text{where } D_i = \begin{cases} 0, d_i \leq 0 \\ d_i, d_i > 0 \end{cases}$$

Our example yields a noise penalty component of 0.3125.

- II.** From our example we would be missing one A and one C. The absence component of the metric is calculated in a manner similar to the noise component using the following formula, where  $s$  is the number of tokens in the source sequence, and  $m_i$  is the token absence weight. Note that dividing the summation by  $s$  normalizes this component to a range between 0.0 and 1.0.

$$\frac{1}{s} \sum_{i=1}^a D_i m_i, s \geq 0 \quad \text{where } D_i = \begin{cases} d_i, d_i > 0 \\ 0, d_i \leq 0 \end{cases}$$

Our example yields a missing penalty component of 0.25.

- III.** Sequence position is the relative location of the tokens as they are laid out in a sequence of tokens. The source string is assumed to maintain the positional information for the comparison. To compute the sequence position component of

the metric, a cost is calculated for each token in the source string as they appear in order from left to right, with the cost being the number of positions off from the expected location in the target string. The expected location is determined by locating the tokens in the source sequence in order as they occur, then finding their relative order in the target. If a token is not found in the target then the absent token is ignored, the expected position is incremented, and the next token from the source is considered. The following table is from our example, where A' denotes the second A found in the source token sequence. The maximum possible offset for any given token is computed as  $\text{MAX}(l-e-1, e)$ , where  $l$  is the length of the target sequence and  $e$  is the expected position of the source token within the target sequence. The sequences are indexed starting with 0 at the first position.

**Table 6: Offset calculations between ABCA and BBAD**

Token	Expected Position	Located at Position	Offset From Expectation	Maximum Possible Offset $M_i$
A	0	2	2	3
B	3	1	2	3
C	2	Absent	0	0
A'	3	2	1	3

The sequence position component is calculated by the following formula where  $a$  is the number of tokens in the source sequence, and  $p_i$  is the token's position weight. The variable  $o_i$  is the number of positions offset from the expected position in the target sequence from where the token is actually found. If the token is missing in the target sequence then the offset is 0. The variable  $M_i$  is the maximum possible offset from the expected position for a token in the target sequence or 0 if the token is missing in the target sequence.

$$\frac{\sum_{i=1}^a o_i p_i}{\sum_{i=1}^a \text{SumMaxOff}_i}, \text{SumMaxOff}_i = \begin{cases} 0 & \text{if } M_i = 0 \\ M_i & \text{if } M_i > 0 \end{cases} \text{ where } \text{SumMaxOff}_i = \sum_{i=1}^a M_i$$

Our example yields a sequence position component of 0.5556.

The overall similarity value is calculated by the formula below. The metric is scaled from 0.0 to 1.0, with 1.0 representing a perfect match and 0.0 representing no similarity.

**Table 7: Penalty component weighting**

Metric Component	Weight
noise	.75

absence	.25
sequence position	1.0

$$1.0 - ((nw_1 + mw_2 + pw_3) / 3)$$

The similarity value from our example is

$$1.0 - ((0.075 * 0.3125 + 0.25 * 0.25 + 1.0 * 0.5556) / 3) = 0.7158.$$

#### 4.6.5 Conclusion

The problem of determining how closely two sequences match is highly dependent on individual interpretation. Trying to mimic human behavior in measuring sequence similarity appears to not be cost effective. The Token Sequence Weighted metric provides a flexible method that a user could adapt to meet current analysis needs. An additional benefit of the algorithm is that it can be implemented such that it can calculate similarity in sequences of data structures much more complex than strings of characters.

In the next section, we discuss the Pattern Discovery Tool, which is a JAVA implementation of many of the algorithms and techniques described in this section. We also present our findings which resulted from applying this tool to several domains, including unattended monitoring sequences (bunker event data) and robot navigation sensor and motion event sequences.

# 5 The Pattern Discovery Tool

## 5.1 Introduction

The Pattern Discovery tool is a prototype tool developed during the course of the Fuzzy Data Mining LDRD. The purpose of the tool is to allow an analyst to view token symbol sequences that represent sequences of events in chronological order as they were logged in a database. The purpose of the analysis on these event sequences is to discover patterns in the data. These patterns may be exact matches or somewhat fuzzy matches. The visualization aspect of the tool is used to display patterns in their context. The user can also manipulate the sequence by eliminating or replacing recurrent patterns. As the tool discovers patterns it provides statistical measures to provide a means for assessing the relative significance of the patterns.

## 5.2 Tool Components

The tool has several visual components. The edit area is where the token sequence is displayed. This area is a functioning editor that allows the tokens to be modified in an ad hoc mode by the user. When a new file, token sequence, is read into the tool, the new sequence overlays the old one in the editor window. On the right side of the tool is the pattern list. This is the list of patterns that the pattern discovery algorithm located within the token sequence. The definition of a pattern for this list is any subsequence of tokens up to the Pattern Size that occurs more than once in the whole sequence. During the pattern discovery process information for calculating statistics are gathered. The patterns are displayed with an initial statistic. The initial statistic is  $MP(ABC)$  which is the probability of finding the particular pattern at any location within the entire sequence. If a user selects a row in the table by clicking on it, the pattern is selected and displayed in the Current Pattern field. The Current Pattern field will be used in various tool functions. The window in the lower left portion of the screen provides information about the actions that have occurred. Figure 8 displays the tool's main screen.

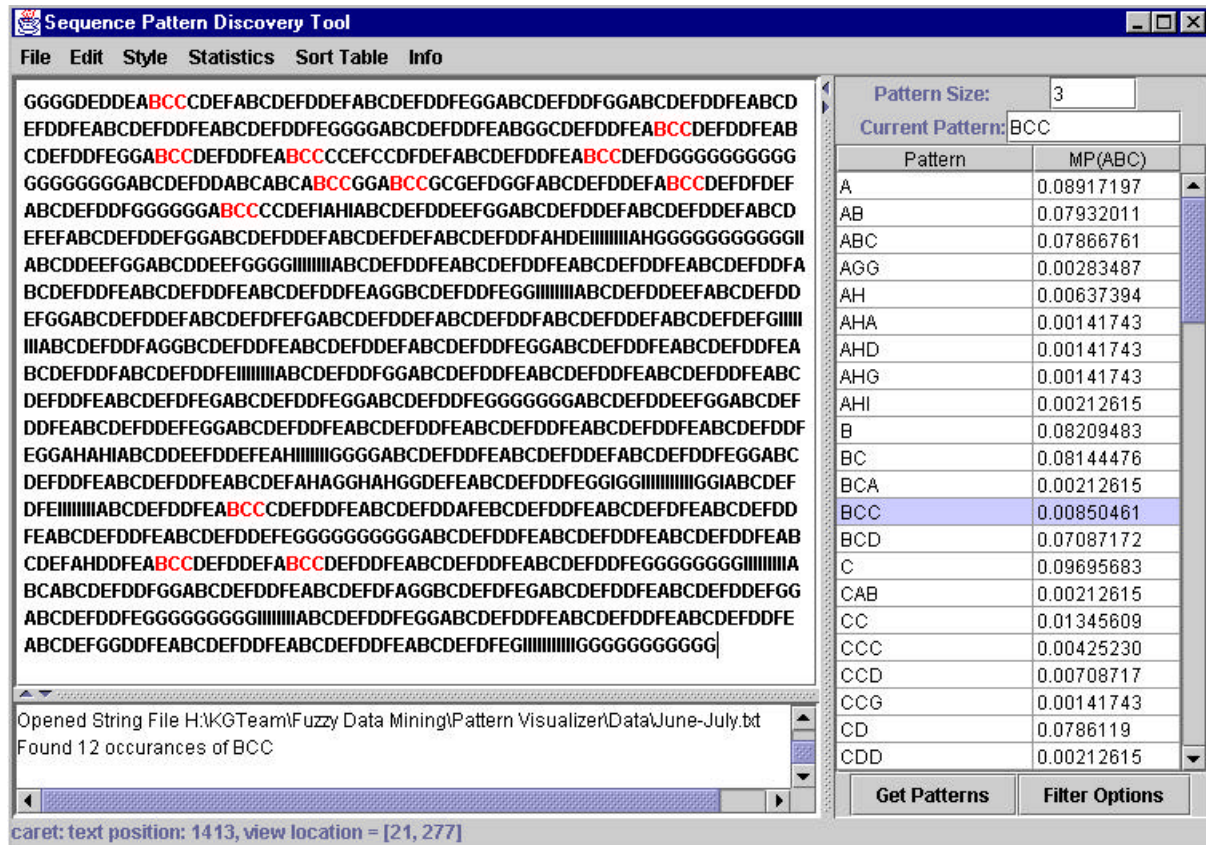


Figure 8: The main *Sequence Pattern Discovery Tool* screen

### 5.3 Editing Sequences

As described earlier the edit window is a functioning editor that allows a user to add or delete tokens just like regular text. All or any portion of the sequence can be selected, highlighted, replaced, cut or pasted. There are some additional edit capabilities that are located in the Edit drop down menu that are especially useful in analyzing token sequences. See Figure 9. Pattern Consolidation and Pattern Replacement operate on the selection in the Current Pattern field. Pattern Replacement simply replaces every occurrence of the selected pattern with another token as provided by the user. Pattern Consolidation replaces repeating sequences of the selected pattern with a single replacement token or sequence of tokens. Hold Current String allows the user to put the current sequence in a buffer that can be put back into the editor later in the session. Reanalyze Current String will run the sequence in the editor back through the pattern discovery process. This is necessary if edits have occurred. The sequence can also be written to a file by using the File drop down menu and selecting Save String to File. The Style drop down menu allows the text of the sequence to be displayed in different styles. Style changes are not persistent at this time and have limited utility.

Undo style change
Redo
cut-to-clipboard
copy-to-clipboard
paste-from-clipboard
select-all
Pattern Consolidation
Pattern Replacement
Hold Current String
Reanalyze Current String

Figure 9: The *Edit* drop down menu

## 5.4 Pattern Statistics

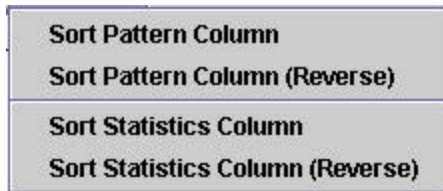
The Statistics drop down menu, Figure 10, allows the user to choose the pattern statistic that is displayed with each pattern in the Pattern List. When a statistic option is selected it is recalculated and displayed for each pattern in the list. The definition for each of these statistical measures was discussed previously in section 4.5.3.

Count
MP(ABC)
MF(ABC)
MP(A)MP(B)MP(C)
MF(MP(A)MP(B)MP(C))
MP(A)MP(BC)
MP(AB)MP(C)
RP(ABC)
RF(ABC)
RP(A)
RF(A)

Figure 10: The *Statistics* drop down menu

## 5.5 Table Sorting

The *Table Sort* drop down menu, shown in Figure 11, allows the user to sort the pattern list table by the pattern in lexical order in either ascending or reversed order, or by the selected statistic in ascending or descending order.

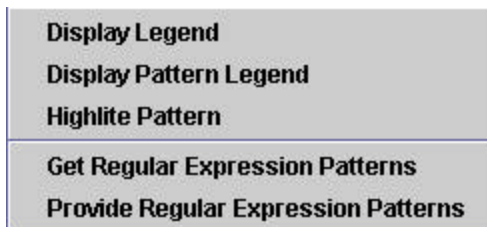


**Figure 11: The *Sort Table* drop down menu**

## 5.6 Pattern Information

The "Info" drop down menu, Figure 12, allows the user to gain additional insight about patterns.

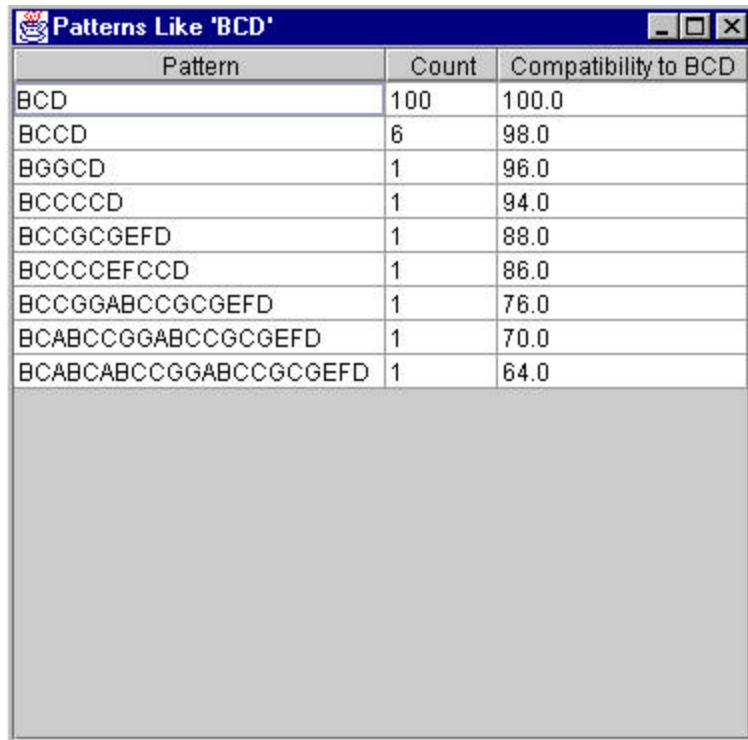
- ?? Display Legend displays a dialogue that lists the definition of all the distinct tokens found in the sequence.
- ?? The Display Pattern Legend dialogue shows the meaning of any specific pattern.
- ?? Highlite Pattern will change the color of the text for each instance in the sequence of the selected pattern shown in the Current Pattern field. Figure 8 shows the pattern BCC highlighted in red.



**Figure 12: The *Info* drop down menu**

The other two menu items Get Regular Expression Patterns and Provide Regular Expression Patterns direct the system to discover other patterns that may be similar but not exactly like a previously detected pattern. Get Regular Expression Patterns will generate a set of regular expressions from the pattern that is selected and shown in the Current Pattern field. The matching patterns are displayed in the Patterns Like ... window. See Figure 13. You will note that all the patterns begin and end with the beginning and ending tokens of the selected pattern. This is a result of the regular expressions that were generated. It is assumed that first and last tokens of the selected pattern are significant. The patterns displayed are a union of all the results from executing each of the regular expression permutations. In the table two numbers are also shown, first the count of the number of instances of the pattern in the sequence, secondly a compatibility measure, explained in Section 4.5.3, quantifying how close or similar the detected pattern is to the original. As implemented, the algorithm assumes all weights are one. The tool could be modified easily to accept user input for the weights.

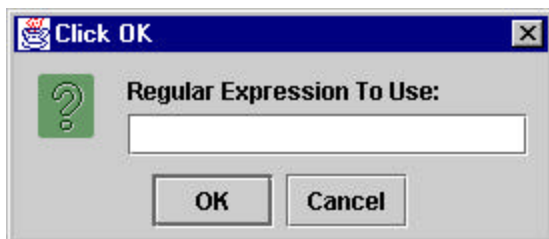




Pattern	Count	Compatibility to BCD
BCD	100	100.0
BCCD	6	98.0
BGGCD	1	96.0
BCCCCD	1	94.0
BCCGCGEFD	1	88.0
BCCCCFCCD	1	86.0
BCCGGABCCGCGEFD	1	76.0
BCABCCGGABCCGCGEFD	1	70.0
BCABCABCCGGABCCGCGEFD	1	64.0

**Figure 13: The *Patterns Like ...* window**

The last item in the "Info" drop down menu is Provide Regular Expression Patterns. This option allows a user to enter a regular expression directly in a dialogue. See Figure 14. When the user requests patterns from a regular expression that he provided, the Patterns Like ... window displays only two columns, the pattern and the count of occurrences in the sequence. See Figure 15. The Patterns Like ... window can be used just like the pattern list table in the main window to select a pattern for the Current Pattern field.

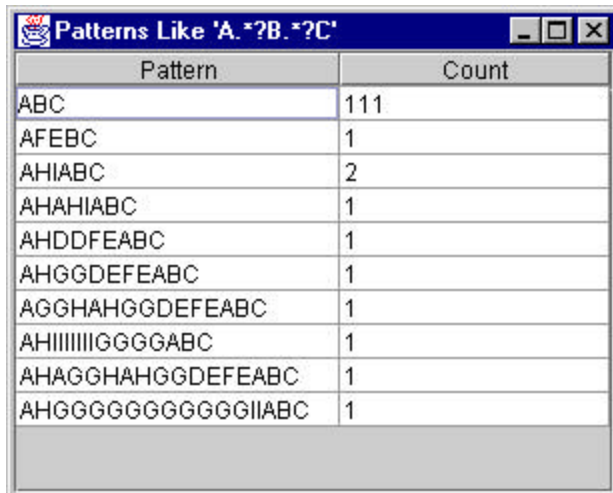


Click OK

Regular Expression To Use:

OK Cancel

**Figure 14: The *Regular Expression Entry* dialogue**



Pattern	Count
ABC	111
AFEBC	1
AHIABC	2
AHAHIABC	1
AHDDFEABC	1
AHGGDEFEABC	1
AGGHAHGGDEFEABC	1
AHIIIIIIIGGGGABC	1
AHAGGHAHGGDEFEABC	1
AHGGGGGGGGGGGGIIABC	1

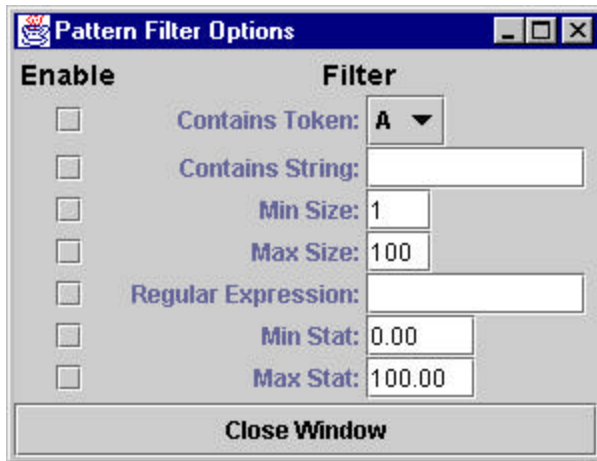
**Figure 15: The *Patterns Like ...* window -- user provided regular expression**

## 5.7 Pattern List Functions

There are four functions that a user may employ when dealing with the pattern list. The ability to sort the list and to select a pattern for the Current Pattern field has previously been discussed.

You will note that at the top of the pattern list of Figure 8 is the Pattern Size field. The pattern discovery algorithm uses this field to determine what the largest pattern size will be. Increasing this number and pressing the Get Patterns button, at the bottom of the list, will invoke the discovery process to search for patterns of larger sizes in the sequence up to the specified size.

The Filter Options button provides choices for filter the list. Pressing this button displays the Pattern Filter Options window, Figure 16. The filter provides a great deal of flexibility in which pattern are shown.



**Figure 16:** The *Pattern Filter Options* window

## 5.8 Applications

### 5.8.1 Unattended Monitoring Data

We obtained a number of data sets from unattended monitoring systems fielded by the International Security Programs Center. Most of the data sets were taken from a simulated storage bunker here at Sandia. This bunker was instrumented with a door switch, two motion detectors, two video cameras, and 10-20 “T-1” multi-sensor item monitors. The T-1s can be attached to items of interest (i.e., containers) with a fiber optic seal and they report seal status, motion, and temperature of the item. The T-1s have two modes: normal and transportation. In normal mode, all of the programmed sensors are active. In transportation mode, the motion sensor does not report, since motion is expected in the transportation process. In order to provide monitoring of the item while it is outside the domain of a monitoring system (i.e., during transportation), the T-1s buffer the 100 most recent events. A buffer dump can be requested at any time the T-1 is in transportation mode. The T-1s also report state-of-health information including battery voltage. The monitored items can be logged in and out of the system. The monitoring system periodically polls the T-1s and records State-of-Health information.

#### *General Observations*

Our initial analysis of data from the simulated storage magazine is based on a simple time-flattened tokenization of the events based on the event types. The tokenization scheme is shown in Table 8.

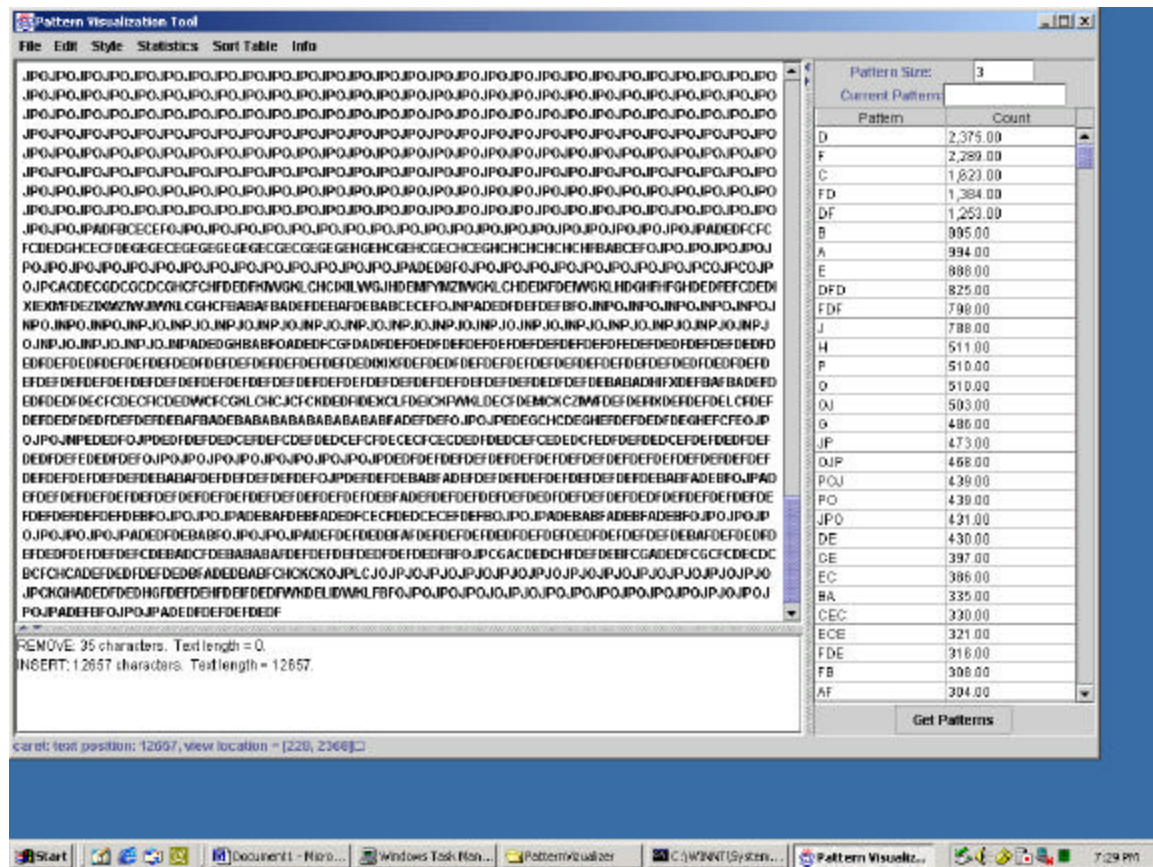
**Table 8:** Event tokenization scheme for initial analysis

Event Type	Token	Description
Door Open	A	Facility door opened
Door Close	B	Facility door closed
Motion Start	C	T-1 Motion started
Occupied	D	Volumetric motion sensor

		active
Image Trigger	E	Camera triggered to capture an image
Unoccupied	F	Volumetric motion sensor inactive
Seal Open	G	Fiber optic seal on a T-1 opened
Seal Close	H	Fiber optic seal on a T-1 closed
T1 Login Started	I	Login process started
T1 Temperature	J	Temperature from T-1
T1 Into Transportation Mode	K	T-1 placed in transportation mode
T1 Out of Transportation Mode	L	T-1 placed in normal mode
T1 Logout Started	M	Logout process started
T1 SOH Poll Failure	N	T-1 failed to respond to SOH poll request
T1 SOH Poll- All Start	O	T-1 poll sequence started
T1 SOH Poll- All Complete	P	T-1 poll sequence completed
Scene Change Image	Q	Image triggered by scene change filter
T1 Poll Failure	S	T-1 failed to respond to poll request
Missing Multiple Tamper Events	T	The counter for case tamper events incremented by more than one event.
RF Interference	U	Indicator that RF interference is present (disrupts communications to T-1s)
No RF Interference	V	Clear event for previous event.
T1 Login Ended	W	Login data received at DCC
T1 Login Cancelled	X	Login process cancelled
T1 Logout Cancelled	Y	Logout process cancelled
T1 Logout Ended	Z	Logout process completed

The data were tokenized and all repeating tokens were condensed to a single token of the same value (e.g., JJJJJJ was replaced with J). The Pattern Discovery Tool was used to iteratively analyze the data to see which patterns occurred frequently and how well those

patterns matched our understanding of the operation of the monitoring system. A typical screen is shown in Figure 17.



**Figure 17: The main *Sequence Pattern Discovery Tool* screen for bunker analysis**

The frequently occurring subsequences are sorted by frequency of occurrence. Note that there are a number of two or three token sequences that occur frequently. We will comment here on some of the frequent patterns.

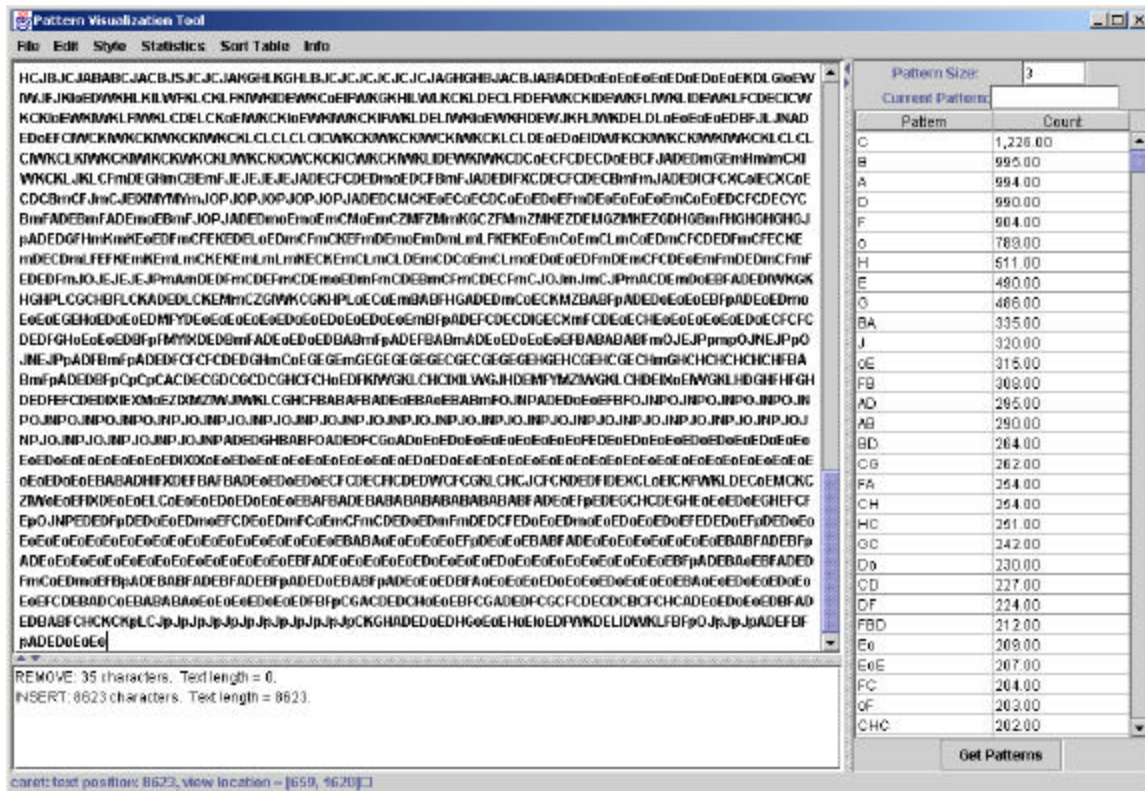
The two-token patterns include FD (or DF), CE (or EC), DE, and BA. It turns out that each of these represents an expected pattern in the event data. Note that DF occurs nearly as many times as FD (1253 vs 1384). Often, the short patterns repeat many times, making it difficult to decide which of the two possible combinations (e.g. FD or DF) is more fundamental. Our observation is that the more frequent pattern is generally the more fundamental pattern, but we still use domain knowledge to make the final decision. The FD combination represents an Unoccupied, Occupied event sequence from a volumetric motion sensor in the facility. The pairing is common because the motion sensor resets if it detects no motion for some fixed time (e.g., 30 seconds) and retriggers if there is additional motion. The FD pairing is more frequent because in some cases other events related to the motion occur after the Occupied event.

The CE and DE combinations indicate triggering of an image based on either a Motion event (from one of the T-1 item monitors) or from a volumetric motion sensor. The cameras in the facility are triggered to capture images when someone enters the facility, moves about within the facility or when the item monitors indicate an item is being handled in some way.

The BA combination represents a Door Close, Door Open event sequence. This sequence is much more frequent than the AB sequence because the door generally remains open when the facility is occupied and there are a number of events related to the activity in the magazine between the open and close events.

OJP is a frequently occurring three-token sequence. This sequence consists of the T-1 SOH Poll- All Start, T1-Temperature, T1-SOH Poll- All Complete events and represents a SOH poll of all the T-1s in the facility. The T-1 Temperature is only event from the SOH message that is tokenized in this scheme. Recall that initially, repeating tokens were condensed to a single token, so the J in the OJP sequence actually represents many temperature events. The large number of repeated poll events represents a period of minimal activity at the facility in which the only events were the polls.

Once the poll events were identified, the OJP sequence was replaced with p for poll and repeating poll events were condensed. In addition, the CE sequence was replaced with m for motion and consolidated and the DF sequence was replaced with o for occupied and consolidated. The result is shown below in Figure 18. The replacement and consolidation is useful for allowing other patterns to become more apparent in the data.



**Figure 18: The updated main *Sequence Pattern Discovery Tool* screen. The screen has been updated to reflect token replacement and consolidation described in the text.**

This analysis was intended as a simple validation test to show that patterns we expected to find in the data (e.g., polls, image triggers, pairing of events from sensors like the doors and motion sensors) could be found by this approach and would appear to be significant based on frequency measures.

It became clear at this point that there were at least three classes of events in the storage facility that would be worth pursuing independently: *poll events*, *logins/logouts*, and *other activity*. Each set was tokenized separately to look at more specific questions. A partial analysis of login data is given below.

### Login Process Analysis

We next used the Pattern Discovery Tool to identify patterns in the login process. In other analyses of the storage facility data we had observed a number of variations in the expected sequence of events in these processes, and we wished to identify all of the variations. The login process is explained below:

- Initially the T-1 is in transportation mode.
- A bar code reader is used to indicate that a login is beginning.
- The bar code reader is then used to scan the T-1, the container of interest, and the container's location information.
- The information is transmitted to the monitoring system via RF communications.



5. The monitoring system requests a buffer dump from the T-1 and records the events.
6. The T-1 is switched out of transportation mode.
7. Logins may be cancelled using an appropriate bar code with the bar code scanner

We tokenized the data using a time-flattened scheme based on the event types associated with T-1s and logins. These events are listed in Table 9.

**Table 9: Event tokenization scheme for login process analysis**

Event	Token	Comment
Case Tamper Active	A	Indicates T-1 case has been tampered with
Case Tamper Inactive	B	Indicates T-1 case tamper indicator is in its normal state
Missing Event	C	T-1 Event counter incremented 2 or more above previous value. Indicates possible missed event from a T-1
Seal Close	D	Fiber optic seal on a T-1 closed
Seal Open	E	Fiber optic seal on a T-1 opened
T1 Buffer Dump End	F	T-1 message indicating end of a buffer dump
T1 Buffer Dump Fail	G	DCC message indicating T-1 buffer dump did not occur
T1 Buffer Dump Poll Failure	H	DCC message indicating T-1 did not respond to poll request
T1 Buffer Dump Start	I	T-1 message beginning a buffer dump
T1 Buffer Read Done	J	DCC signals completion of T-1 buffer read
T1 Buffer Read Start	K	DCC requests T-1 buffer dump
T1 Buffer Read Stop	L	DCC signals completion of T-1 buffer read
T1 Into Transportation Mode	M	T-1 placed in transportation mode
T1 Login Cancelled	N	Login process cancelled
T1 Login Ended	O	Login data received at DCC
T1 Login Started	P	Login process started
T1 Logout Cancelled	Q	Logout process cancelled
T1 Logout Ended	R	Logout process completed
T1 Logout Started	S	Logout process started
T1 Out of Transportation Mode	T	T-1 placed in normal mode
Unknown Event	U	The T-1 event counter indicates at least one event occurred, but it is not possible to infer what caused the event from the state of the T-1 sensors.



The analysis produced a number of interesting observations. First, there was a large number of Buffer Dump End events – 14,538 of the 15434 events. Buffer Dump End events are recorded by the T-1 at the end of each buffer dump and are put in the buffer. If a T-1 is re-used many times, it is possible to have a number of Buffer Dump End events in a given buffer. We decided that the number of Buffer Dump End messages was not an important distinguishing characteristic and consolidated any repeating sequences into a single event.

Logins normally begin with a fixed sequence of events: Login Start, Login End, Buffer Read Start, Buffer Dump Start. These events are tokenized as POIK. Logins normally end with another fixed sequence of events: Buffer Dump End, Buffer Read Stop. These events are tokenized as FL. In between, there may be any number of other T-1 events (M, A, B, U, T, D, E, C) in any order. A few example normal login sequences are shown in Table 10. The expected initial and final sequences are separated from the middle sequence to highlight the similarities and differences in the patterns.

**Table 10: Frequency of selected normal login patterns**

Pattern	Frequency
POIK MABM FL	8
POIK MABMBM FL	2
POIK MAMBAB FL	5
POIK MAMB FL	6

There were 67 Login Started events. Of these, the frequency and interpretation of some of the more common patterns are given in Table 11. The asterisk is used to indicate any number of T-1 events in any order. “Normal” login events do occur frequently, but there are a number of deviations. As shown in the second row, one common deviation is that there is some problem completing the buffer dump. In these event sequences, the events between the Buffer Read Start and the Buffer Dump Poll Failure message usually relate to additional attempts to read the buffer rather than actual sensor events like Seal Open or Seal Close. Another common deviation from the normal login sequence is canceling the login as shown in the third and fourth rows.

**Table 11: Frequency of selected login patterns**

Pattern	Frequency	Interpretation
POIK * FL	31	Normal Login
POI * H	5	Unable to dump buffer
PN (login start, login cancelled)	8	Login Cancelled
P * N	3	Abnormal login cancelled

Categorizing the remaining login attempts will require more work. A large number of the remaining attempts show a disruption in the initial sequence – that is, one or more events

inserted between the P and O, the O and I, and/or the I and K. In addition, there appear to be occasions where logins overlapped. An example of this is given below:

**POIP****KmBMFL***OIKmBMFL*

where the apparently overlapping sequences are distinguished by bold and italic text and the “m” indicates some number of T1 Into Transportation Mode events. It is possible the disruption in the initial sequence is due to intermingling of events from different T-1 in the monitoring system. However, the apparent overlap of login sequences seems very odd based on our knowledge of operational procedures at this test bed. We need to go back to the raw data and possibly modify the tokenization scheme to clarify these issues.

### 5.8.2 Network Intrusion Detection

Computer security personnel tasked with the network intrusion detection problem face many similar issues in mining sequences of events in data. While there is no formal data mining project in place, members of the Computer Security group at Sandia National Laboratories are working on methods to identify and detect patterns in network traffic that indicate network intrusion or other suspicious activity. Commercial network monitoring tools do provide some assistance to this end, but they generally lack the flexibility needed to develop proactive early detection schemes. Most of these tools are based on the use of signatures (predefined data sequences) which are scanned for in network traffic and data stores. Such tools are limited to pattern detection, that is, the identification of patterns in data that have already been predefined, rather than pattern discovery.

What is desired is a tool to discover new patterns that deviate from normal expected behavior so that intrusion attempts and other unauthorized activity can be detected, even though a pre-existing pattern (signature) for that activity was not previously known [34]. For example, the Code Red worm [35] recently infected many computer systems throughout the world. This worm spreads by generating permutations on a given network address to discover new networks to attack. This behavior can be characterized by the type and quantity of network traffic that it generates. A pattern discovery tool that is able to identify such anomalies in network traffic would be a valuable asset in detecting new generations of worms and viruses that propagate in a similar way. With such a tool, new malicious activity could be detected early, before it propagates within the organization and causes significant damage.

A short demonstration of the Pattern Discovery Tool was well received when shown to Computer Security personnel at Sandia. Of particular interest was the tool’s ability to detect and isolate new patterns found in a user-defined token sequence. Using a simple dataset, we were able to isolate an interesting set of communications between a web server and a firewall, using the statistical measures and filtering capabilities of the tool. This demonstration also led to suggestions of how to extend the tool to facilitate the ability to drill-down into underlying data by selecting a token or sequence of tokens. For example, information fields contained in the headers of packets in network traffic are not always needed or used. Unauthorized communication can take place by sending packets with benign payload data and encoding the real message in the unused header fields of

the packets. Because the header fields are relatively small, even a modest message communicated in this fashion would require many packets to encode. The Pattern Discovery Tool could be used to identify network traffic that follows this pattern. Once a pattern is found, then a useful ability would be to examine the packet headers and data associated with the pattern to determine if any anomalies exist. Going one step further, all occurrences of the pattern could be isolated in the tool and re-tokenized based on some field in the associated data, perhaps from an attached database. Then patterns in this new context could be found to further process the data and gain insights into the true nature of the network traffic activity. A key concept of the Pattern Discovery Tool is that the user has direct control of the type and amount of information displayed.

For the third example application, we obtained data from another project that is developing cognitive models for decision support systems [36]. The data consisted of 42 data sets, each being one of six runs for seven different robots. The data sets are sequences that are periodic snapshots of the states of the robots as they are performing some task. Each record in the file represents the state at a given time. The first field is a time value starting at 0 and incrementing at .05 time units. The remainder of the record is a vector of 40 real values presumably representing the robot's state.

0.11 17.0035 0  
 0.05 11 17.01 00 35 0  
 0.1 10.999 17.02 00 0.1 35 0  
 0.15 10.994 17.0697 00 0.1 35 0  
 0.2 10.994 17.0697 00 0.2 35 0  
 0.25 10.9841 17.1187 00 0.2 35 0  
 0.3 10.9841 17.1187 00 0.3 35 0  
 0.35 10.9693 17.1665 00 0.3 35 0.10 0.1 10 0.10 0.1 10 0.1 10 0.1 11 1 1 0.1 10 0.1 10 0.1 10 0.1 10 0.1 1

It appears that each of the six runs for a given robot is a training run for that robot and that each robot is trying to accomplish the same task. Therefore, all of the 42 data sets were concatenated into a single data set and run through the discovery process. Enough equivalent patterns across the combined data set were found to indicate commonality between the various runs; there were enough differences to indicate the runs were unique.

**Table 12: Runs within the data set**

1	403	1
404	806	2
807	1209	3
1210	1612	4
1613	2015	5
2016	2419	6

**Table 13: Frequent patterns found across robot runs**

Pattern #	Starting Records	Pattern Length
1	1, 1613	263
2	1, 404, 807, 1210, 1613, 2016	7
3	1, 807, 1613	55
4	1, 807, 1613, 2016	26
5	232, 1844, 2247	32
6	232, 2247	172
7	308, 1517, 2323	75
8	359, 1568, 1971, 2374	24
9	359, 1971	52
10	359, 1971, 2374	45
11	436, 2048	193
12	468, 871, 2080	140
13	597, 1000, 1403, 2209	11
14	597, 1403	105
15	597, 1403, 2209	32

Analysis of the data shows that all the runs are exactly the same for the first 7 records and this is the only sequence in common among all the runs. Two runs end the same. When there is any commonality between two or more runs they occur at the exact same point into the run.

At this point, we do not have sufficient information to examine the significance of the discovered patterns, but this example was an excellent test of the ability of the pattern detector to discover patterns in a sequence of arbitrary object representations.

## 6 Summary and Future Work

The data we are interested in analyzing is a mixture of analog (i.e., continuous-valued data) and discrete events. As a result, we explored techniques for pattern matching and pattern discovery in both continuous and discrete events. For pattern matching in continuous data, we studied the use of Dynamic Time Warping (DTW) [6][7][8][9][10] or Hidden Markov Models. We had a limited set of analog data to work with for this aspect of the project and were not able to proceed beyond the initial investigatory steps. However, based on our analysis of the DTW and HMM algorithms, we would focus future efforts on Derivative Dynamic Time Warping (DDTW).

In the second half of the project, we explored techniques for pattern matching and pattern discovery in discrete event data. We developed a Pattern Discovery Tool based on adaptations of the A-priori [11] and GSP (Generalized Sequential Pattern mining) [12] algorithms. We then used the tool on three different application areas – unattended monitoring system data from a storage magazine, network intrusion detection, and analysis of robot training data.

As described earlier, the tool is prototype; even so it demonstrates several capabilities. First and foremost it demonstrates the ability to discover patterns in a symbol sequence and provided a visual mechanism to understand those findings. Second it demonstrates the ability to detect subsequences that are similar to a baseline pattern and provide a metric of that similarity.

The concepts behind the tool certainly exhibit promise and there are a number of enhancements suggested by individuals who have seen demonstrations of the tool, as well as possible uses. The tool shows promise in analyzing network intrusion data, bunker sensor data, and state sequences from robot test runs. Other possibilities may include non-temporal data such as textual patterns or genome sequences.

Some possible enhancements are to allow for event tokens of more than a single token in length or possibly variable length token sets. Also the ability to derive patterns from a sequence data set rather than just a token sequence would prove very useful. This functionality could include the ability to drill-down into the supporting data represented by the token. For example a sequence of TCP/IP packet headers could be shown as either tokens which represent host or destination addresses, or as the list of addresses themselves. From any individual element in the sequence a user could display the underlying packet data. The analyst could direct that certain elements be hidden rather than edited out. From the underlying data structure different elements could be used as the display elements for visualization. For such a mechanism to work properly and still be able to find similar subsequences a method like regular expressions would need to be devised which operates on arbitrary data structures in preference to single letter tokens. A state machine engine may prove useful for achieving this. Greater use of highlighting, coloring and styles could be employed to highlight different types of patterns, overlaying patterns, etc.

## 7 References

- [1] J. M. Brabson, "Finite State Machine Analysis of Remote Sensor Data", Proceedings of the 40<sup>th</sup> Annual Meeting of the Institute of Nuclear Materials Management, Phoenix, AZ, July 1999.
- [2] S. M. DeLand, J. M. Brabson, J. D. Smith, T. I. Jaramillo, S. M. Spaven, "Analysis of Unattended Monitoring System Data Using Knowledge Generation", Proceedings of the 41<sup>st</sup> Annual Meeting of the Institute of Nuclear Material Management, New Orleans, LA, July 2000.
- [3] J. M. Brabson and S. M. DeLand, "Knowledge Generation," presented at the European Safeguards Research and Development Association 3rd Workshop on Science and Modern Technology for Safeguards, Tokyo, Japan; November 2000.
- [4] B. Thuraisingham, *Data Mining: Technologies, Techniques, Tools, and Trends*, CRC Press, Boca Raton FL (1999).
- [5] A. A. Freitas, and S. H. Lavington, *Mining Very Large Databases with Parallel Processing*, Kluwer Academic Publishers, Boston (1998).
- [6] D. J. Berndt and J. Clifford, "Finding Patterns in Time Series: A Dynamic Programming Approach." In *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. AAAI Press / The MIT Press: 229-248. 1996.
- [7] E. J. Keogh and M. J. Pazzani, "A Simple Dimensionality Reduction Technique for Fast Similarity Search in Large Time Series Databases." In the *Fourth Pacific- Asia Conference on Knowledge Discovery and Data Mining*. Kyoto, Japan. 2000.
- [8] E. J. Keogh and M. J. Pazzani, "Scaling up Dynamic Time Warping to Massive Datasets." In *3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*. Prague. 1999.
- [9] E. J. Keogh and M. J. Pazzani, "Derivative Dynamic Time Warping." Unpublished. 2000.
- [10] T. Oates, L. Firoiu, and P. R. Cohen, "Using Dynamic Time Warping to Bootstrap HMM-Based Clustering of Time Series." In *Sequence Learning: Paradigms, Algorithms and Applications*, R. Sun and L. Giles, editors. Springer-Verlag. 2000.
- [11] R. Agrawal & R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487-499, Santiago, Chile, Sept. 1994.
- [12] C. Bettini, S. Jajodia & S. Wang. *Time Granularities in Databases, Data Mining, and Temporal reasoning*. Springer, 2000.
- [13] G. Karypis and V. Kumar. "Scientific Data Mining".  
<http://www.ca.sandia.gov/ASCI/cgi-bin/sdmframedisplay.cgi/ASCI/sdm/PatternDiscovery.html>
- [14] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: An efficient data clustering method for large databases". In *Proceedings of the 1996 ACM-SIGMOD International Conference on Management of Data*, Montreal, Quebec, 1996.
- [15] M. V. Joshi, G. Karypis, and V. Kumar. "Universal formulation of sequential patterns". Technical Report 99 1, Department of Computer Science, University of Minnesota, Minneapolis, 1999.

- [16] L. O. Hall, K. W. Bowyer, N. Chawla, T. Moore, Jr., and W. P. Kegelmeyer. "AVATAR – Adaptive Visualization Aid for Touring and Recovery". Technical Report SAND2000-8203, Sandia National Laboratories, Albuquerque, NM, January 2000.
- [17] J. R. Quinlan. "Improved Use of Continuous Attributes in C4.5". *Journal of Artificial Intelligence Research*, 4, pp. 77-90, March 1996.
- [18] J. R. Quinlan. "C4.5 – Programs for Machine Learning". Morgan Kaufmann, San Mateo, CA, 1993.
- [19] E. Bradley, N. Collins, and W. P. Kegelmeyer. "Feature Characterization in Scientific Datasets". *Proceedings of the International Workshop on Intelligent Data Analysis*, September 2001.
- [20] M. C. Miller, J. F. Reus, R. P. Matzke, W. J. Arrighi, L. A. Schoof, R. T. Hitt, and P. K. Espen. "Enabling interoperation of high performance, scientific computing applications: Modeling scientific data with the sets & fields modeling system". In *International Conference on Computational Science (ICCS-2001)*, 2001.
- [21] G. S. Davidson, B. Hendrickson, D. K. Johnson, C. E. Meyers, and B. N. Wylie, "Knowledge Mining with VxInsight: Discovery through Interaction," *Journal of Intelligent Information Systems*, 11(3), November/December 1998, pp.259-285.
- [22] S. Motroni and H. Vanderberg. "MineSet Enterprise Edition User's Guide for the Windows Client", Document Number 007-4005-002, Silicon Graphics, Incorporated, Mountain View, California. See also <http://www.sgi.com/software/mineset/>.
- [23] B. Hendrickson, D. Johnson, B. Wylie, G. Davidson, C. Meyers, H. Small, and D. Pendlebury. "Navigation Science", *Proceedings of the Symposium on Advanced Information Processing and Analysis (AIPA-97)*. Tyson's Corner, Virginia, March 25-27, 1997. p. 34.
- [24] N. H. Irwin, J. van Berkel, D. K. Johnson, and B. N. Wylie. "Navigating nuclear science: Enhancing analysis through visualization". Technical Report SAND97-2218, Sandia National Laboratories, Albuquerque, NM, 1997.
- [25] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." In *Proceedings of the IEEE*, vol. 77, no. 2: 257-286. February 1989.
- [26] R. Agrawal & R. Srikant. Mining sequential patterns: Generalizations and performance improvements. In *Proc. 5<sup>th</sup> Int. Conf. Extending Database Technology (EDBT'96)*, pages 3-17, Avignon, France, Mar. 1996.
- [27] H. Mannila & M. Salmenkivi. Finding Simple Intensity Descriptions from Event Sequence Data. In *Proc. 7<sup>th</sup> ACM SIGKDD Int. Conf. On Knowledge Discovery and Data Mining*, pages 341-346, San Francisco, California, Aug. 2001.
- [28] B. Liu, W. Hsu & Y. Ma. Identifying Non-Actionable Rules. In *Proc. 7<sup>th</sup> ACM SIGKDD Int. Conf. On Knowledge Discovery and Data Mining*, pages 329-334, San Francisco, California, Aug. 2001.
- [29] Z. Zheng, R. Kohavi & L. Mason. Real World Performance of Association Rule Algorithms. In *Proc. 7<sup>th</sup> ACM SIGKDD Int. Conf. On Knowledge Discovery and Data Mining*, pages 401-406, San Francisco, California, Aug. 2001.
- [30] J. Han & J. Pei. Pattern-growth Methods for Sequential Pattern Mining: Principles and Extensions. In *Proc. 7<sup>th</sup> ACM SIGKDD Int. Conf. On Knowledge Discovery*

and Data Mining - Temporal Data Mining Workshop Notes, pages 47-55, San Francisco, California, Aug. 2001.

- [31] J. E. Friedl, "Mastering Regular Expressions", O'Reilly, Sebastopol, CA (1997).
- [32] J.-I. Aoe, "Computer Algorithms - String Pattern Matching Strategies", IEEE Computer Society Press, pp. 105-110.
- [33] J. Boehm "fuzzy logic" <http://www.personal.kent.edu/~jtboehm/fuzzy.html>, Kent State University, 27 May, 1999.
- [34] Personal communication with Kevin Nauer. Sandia National Laboratories, Albuquerque, New Mexico, August 2001.
- [35] CERT Advisory CA-2001-19. CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania.  
<http://www.cert.org/advisories/CA-2001-19.html>
- [36] Forsythe "Conceptual Design for a Cognitive Architecture with Human-Like Episodic Memory", Sandia National Laboratories, in press.



## 8 Appendix: String Similarity Poll

### Character String Similarity Poll

To the right of each string pair below, write a number between 1 and 10 that indicates how similar you think the two strings are. A 1 indicates very little similarity; while a 10 indicates that the two strings are very similar. Use your own metric for determining similarity.

String Pair	Similarity	String Pair	Similarity
ABCD DEFG		TWEUDHSAOL EUDSA	
ABCD USDABCD		HQWEISFEISVVHQWEISFEISVV DOIUBCPOWIOBYDERUIVYBEWS	
TWEUDHSAOL TWEEDHESAL		ABCD DCBA	
HQWEISFEISVV HQWEISFEISVVHQWEISFEISVV		HIJK HJKI	
AXBXCXDXEX ABCDE		TWEUDHSAOL TEDSOL	
SGITBAFGMAT SGTBFGMT		ABCD ABCCD	
ABCD ABKD		TWEUDHSAOL QWEUDHSAOP	
ABCDE AABBCCDDEE		HQWEISFEISVV HQWEISFELMNOP	
QOUA ZXCM		TWEUDHSAOL HSAOLTWEUD	
MGITHWPA SGITBAFGMAT		AWBOCZDPE ABCDE	

## DISTRIBUTION:

1	MS 0188	LDRD Office, 1030
1	0780	D. G. Adams, 5838
1	0780	L. A. Cano, 5838
1	0780	A. P. Heath, 5838
1	0780	S. Ortiz, 5838
1	0813	S. R. Carpenter, 9327
1	0813	K. S. Nauer, 9327
1	0813	R. A. Suppona, 9327
1	0829	E. V. Thomas, 12323
1	1137	J. L. Mitchiner, 6534
1	1137	K. L. Hiebert-Dodd, 6534
1	1137	J. M. Brabson, 6534
3	1137	J. M. Britanik, 6534
3	1137	G. N. Conrad, 6534
5	1137	S. M. DeLand, 6534
3	1137	C. L. Jenkin, 6534
1	1137	S. D. Kleban, 6534
1	1137	S. J. Starks, 6534
1	1170	R. D. Skocypec, 15310
1	1188	J. C. Forsythe, 15311
1	1213	D. S. Blair, 5301
1	1215	M. A. Grohman, 5326
1	1215	C. D. Croessmann, 5326
1	1361	B. H. Corbell, 5323
1	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 9612
1	0612	Review and Approval Desk, 9612
		For DOE/OSTI